THE EVALUATION OF AN ARM-BASED ON-BOARD COMPUTER FOR A LOW EARTH ORBIT SATELLITE

Gregor Dreijer

Thesis presented in partial fulfilment of the requirements for the Master of Electronic Engineering degree at the University of Stellenbosch

Supervisor: Prof P.J. Bakkes

December 2002

DECLARATION

I, the undersigned, hereby declare that the work contained in this thesis is my original work, and that I have not previously, in its entirety or in part, submitted it at any university for a degree.

ABSTRACT

The use of commercial-off-the-shelf (COTS) components and emerging technologies in satellite systems has become increasingly popular over the past few years. This is mainly due to advances in radiation shielding and system-level reliability improving techniques. The use of a new generation commercial processor in the design of a satellite's on-board computer (OBC) is now considered a feasible option.

The aim of this thesis was to evaluate the use of a commercial grade ARM processor in a *low* earth orbit (LEO) microsatellite on-board computer. The process began with the selection of the most suitable ARM processor for an OBC design. A typical OBC system was developed for the chosen processor, in order to test its functionality and performance in an OBC design.

OPSOMMING

Die afgelope paar jaar het die gebruik van gewone kommersiële komponente en die nuutste tegnologie in satelliet stelsels heelwat toegeneem. Dit kan grootliks toeskryf word aan die vordering in bestralings afskerming en stelselvlak betroubaarheid tegnieke. Die gebruik van 'n nuwe generasie kommersiële verwerker in die ontwerp van 'n satellite aanboord rekenaar (AR) is nou prakties uitvoerbaar.

Die doel van hierdie tesis was om die gebruik van 'n ARM verwerker in 'n lae aardwentelbaan mikrosatelliet aanboord rekenaar te evalueer. Eerstens moes die mees geskikte kommersiële ARM verwerker vir 'n AR ontwerp gevind word. Daarna is 'n tipiese AR stelsel ontwikkel om die verwerker se funksionaliteit en werkverrigting te toets.

CONTENTS

ACKNOWLEDGEMENTS

I would like to thank the following people for their contributions:

- Prof. Bakkes, my supervisor, for his encouragement and guidance
- Francois Retief, for his help and advice, regarding embedded systems programming and the GNU Cross Compiler
- Jacu Vos, for all his technical advice
- My family for their support and motivation

LIST OF ABBREVIATIONS AND ACRONYMS

ADC	Analog-to-Digital Converter
ADCS	Attitude Determination Control System
ALU	Arithmetic and Logic Unit
AMSAT	The Radio Amateur Satellite Corporation
ARM	Advanced RISC Machines
ASIC	Application-Specific Integrated Circuit
CD-ROM	Compact Disc ROM
CMOS	Complementary Metal-Oxide Semiconductor
COTS	Commercial-Off-The-Shelf
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DMA	Direct Memory Access
DRAM	Dynamic Random Access Memory
EDAC	Error Detection and Correction
EDO	Extended Data Out
ESA	European Space Agency
ESL	Electronic Systems Laboratory
FIQ	Fast Interrupt Request
FPGA	Field Programmable Gate Array
FPM	Fast Page Memory
FPU	Floating Point Unit
GCC	GNU Cross Compiler
GPIO	General Purpose I/O
I/O	Input / Output
I^2C	Inter-Integrated Circuit
IC	Integrated Circuit
IHU	Integrated Housekeeping Unit
IrDA	Infrared Data Association
IRQ	Interrupt Request
kB	Kilobyte (1024 bytes)
kbps	Kilobits per second (1000 bits per second)

LCD	Liquid Crystal Display
LED	Light Emitting Diode
LEO	Low Earth Orbit
LVDS	Low Voltage Differential Signalling
MB	Megabyte (1024KB = 1048576 bytes)
Mbps	Megabits per second (1000Kbps)
MCP	Multimedia Communications Port
MIPS	Million Instructions Per Second
MMU	Memory Management Unit
NASA	National Aeronautics and Space Administration
OBC	On-Board Computer
OS	Operating System
OST	Operating System Timer
OTPROM	One Time Programmable Read Only Memory
PC	Personal Computer
PCB	Printed Circuit Board
PCMCIA	Personal Computer Memory Card International Association
PDA	Personal Diary Assistant
PIC	Programmable Interrupt Controller
PLCC	Plastic Leaded Chip Carrier
PLL	Phase Lock Loop
PPC	Peripheral Pin Controller
PROM	Programmable Read-Only Memory
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer
ROM	Read Only Memory
RTC	Real-Time Clock
RTOS	Real-Time Operating System
SAA	South Atlantic Anomaly
SCC	Serial Communications Controller
SCM	System Control Module
SDRAM	Synchronous Dynamic RAM
SEE	Single Event Effects
SEL	Single Event Latch-up

SEU	Single Event Upset
SRAM	Static Random Access Memory
SSP	Synchronous Serial Port
SSTL	Surry Space Technology Limited
SUNSAT	Stellenbosch University Satellite
TID	Total Ionising Dose
UART	Universal Asynchronous Receiver and Transmitter
USB	Universal Serial Bus
VDD	SA-1110 Core Voltage Supply
VDDX	SA-1110 I/O Voltage Supply
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VLSI	Very Large Scale Integration

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Satellite On-Board Computers

A satellite consists of a number of interrelated subsystems, each providing a particular service to the satellite. Although these subsystems usually function independently, they must be properly coordinated for the satellite to operate as a whole. This is the role of the on-board computer (OBC) of a satellite. It is responsible for the overall operational control of a satellite. Figure 1-1 shows the typical subsystems of a satellite.

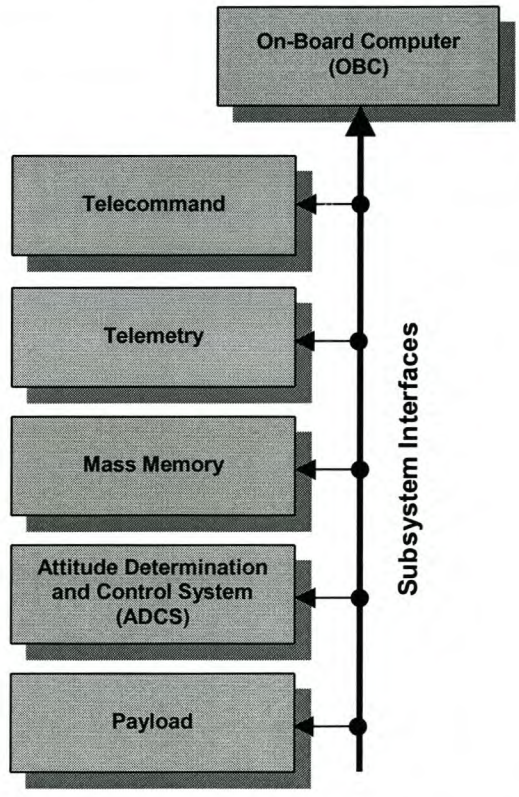

Figure 1-1: Typical Satellite Subsystems

The OBC interfaces to all of the satellite subsystems, to allow it to issue commands to – and request status information from – those subsystems. The exact role of an OBC varies according to the control architecture of the satellite. The control architecture determines how control is distributed between the satellite's OBC and the hardware dedicated to other subsystems. In some satellites, each subsystem is responsible for its own low-level control,

while the OBC only carries out high-level control. In other satellite systems, the OBC is responsible for almost all control actions, where the OBC can be referred to as the Command and Data Handling (C&DH) unit.

Regardless of the control architecture, satellite on-board computer systems are usually designed around a specific processor. The overall performance and functionality of the OBC is therefore highly dependent on the processor used in the design. For this reason, choosing a processor is very important. It has to have sufficient processing performance to carry out the required OBC functions and it must operate reliably in outer space.

1.2 Satellite Design Trends

In the past, satellite systems were designed using only space-qualified technology. This was mainly due to the reliability concerns associated with the space radiation environment. The use of radiation hardened (rad-hard) electronics resulted in very expensive and highly specialized projects.

The high prices of rad-hard devices are a major disadvantage to satellite systems engineers. Furthermore, the range of rad-hard components is very limited. Not all devices are available in rad-hard versions, especially the latest commercial devices from high volume manufacturers. This is because the space community only represents a very small customer segment. Manufacturers will therefore not consider the process and design alterations required to improve the radiation hardness levels of their commercial devices.

The current trend in the design of satellite systems is the insertion of commercial off-the-shelf (COTS) technologies. This direction is clear in the satellite industry, as there were twenty rad-hard component vendors in 1990, and only four today [16]. The use of COTS devices allows satellite systems to take advantage of the latest technology available, which translates to lower costs, higher performance and drastically reduced development times.

Unfortunately, the radiation performance characteristics of these technologies usually show a susceptibility to the space radiation environment. For this reason, the development of a COTS-based system involves adapting existing commercial designs to make them suitable for space use. In such a system, the focus of reliability concerns is shifted from an individual component level to a system level.

The use of COTS devices is especially popular in the design of low-earth orbit (LEO) satellites. This is due to the fact that LEO satellites experience relatively low levels of radiation. The radiation related reliability problems are therefore less significant, allowing the non-radiation hardened electronics to be used more freely.

Using a COTS-based approach to design an on-board computer (OBC) for a LEO satellite is clearly advantageous. It allows a low-cost, high-performance OBC system to be developed, by making use of an up-to-date commercial processor.

1.3 An ARM-Based OBC

The embedded systems market has a strong influence on COTS OBC designs, as these OBCs are usually based on embedded processors. Many OBC designers tend to favour the use of older embedded processors, such as the Intel 80188 and 80386EX processors. The old technology used to manufacture these processors make them less susceptible to the effects of radiation. Unfortunately, these processors limit the performance of the OBCs tremendously. Furthermore, they consume large amounts of power, and may also be discontinued in the near future.

To increase the performance of future OBCs, a new generation of embedded processors will have to be used. There are a large number of high performance embedded processor families currently available: PowerPC, ARM, MIPS and Hitachi SH. However, over the past few years, processor cores developed by Advanced RISC Machines (ARM) Limited have dominated the embedded processor market. This document covers the development and evaluation of a LEO satellite OBC, based on an ARM processor.

1.4 Document Outline

- Chapter 1: **Introduction**
  This chapter provides background information on the thesis subject.

- Chapter 2: **Processor Selection**
  The processor requirements and selection process are covered in this chapter.

- Chapter 3: **Details of the SA-1110**

  The details of the processor, which motivate its use in an OBC design, are discussed.

- Chapter 4: **A Typical OBC Design**

  This chapter covers the hardware design of the OBC system.

- Chapter 5: **Software Development**

  An overview of the ARM programmer's model and the development of the OBC system's software are covered in this chapter.

- Chapter 6: **Tests and Measurements**

  In this chapter, the various OBC test procedures and results are discussed.

- Chapter 7: **Conclusions and Recommendations**

  This concluding chapter summarizes the advantages and disadvantages of an SA-1110-based OBC. It also suggestions of how the OBC system can be improved and identifies areas where further testing of the SA-1110 processor should be carried out.

CHAPTER 2

PROCESSOR SELECTION

The first step in the evaluation of an ARM based on-board computer, was the selection of the most suitable ARM processor. Although there is a large variety of ARM processors available, most of them are designed with a specific commercial application in mind. For this reason, it was unlikely that any ARM processor would be ideally suited for use in a satellite OBC, and some compromises would have to be made.

Since the goal of this project was to provide a low-cost OBC solution, the idea of using an ARM core in a radiation-tolerant ASIC or FPGA was not investigated. Only commercially available discrete ASIC processors were considered.

2.1 Processor Requirements

Before any ARM processors could be compared, it was necessary to compile a list of requirements that the processor should meet:

- Low power consumption
  Power is a very precious resource on a satellite.
- Highest Power Supply Voltage
  To improve radiation tolerance, higher operating voltages are favoured.
- Largest manufacturing technology possible
  With a larger manufacturing process, the silicon density is lower and the processor will be less susceptible to radiation.
- High computational performance
  This will allow for complex software designs and improve system response.
- No permanent Cache Memory
  The on-chip cache memory of a processor is highly prone to radiation related errors.
- Flexible memory controller and I/O interfaces
  The processor should interface easily to memory systems and memory mapped peripherals.

- History of use in Space

  Ideally, the processor should have been successfully flown in space.

Unfortunately, many of the above aspects are interrelated and an improvement in one area often comes at the expense of another. Therefore, these factors had to be considered very carefully. For example, higher voltage levels in a processor generally translate to higher power consumption. However, the higher voltage levels make the processor less susceptible to radiation related errors, as more charge is needed to induce a change in a logic voltage level. The manufacturing process size has similar consequences. Smaller manufacturing processes are usually advantageous, as they are associated with lower power consumption and higher clock speeds. Unfortunately, with smaller process sizes, the device's packing density increases, and this makes them more prone to single event upsets (SEUs). A small process size also increases the potential for radiation charge build-up to bridge silicon gaps and for a high radiation dose to cause breakthrough damage.

2.2 Final Candidates

After an extensive search, the following list of possible processors was drawn up. These processors best fitted the requirements listed above.

Manufacturer	Processor	Clock Speed (MHz)	MIPS[1]	Typical Run Mode Power (mW)	Voltage Supply Core / I/O (V)	Process Size (µm)	Space History
Intel	SA-1100	220	250	550	2.0 / 3.3	0.35	Yes
Intel	SA-1110	206	235	350	1.8 / 3.3	0.35	No
Intel	PXA-250	400	508	500	1.0 / 3.3	0.18	No
Samsung	KS32C50100	50	45	600	3.3 / 3.3	0.35	No
Cirrus Logic	CL-PS7500FE	56	50	800	5.0 / 5.0	0.50	No

Table 1-1: ARM Processor Comparison

[1] Million Instructions Per Second, measured with Dhrystone 2.1

Prior to this thesis, Arno Barnard conducted a local study on possible processors for use in future OBCs [4]. The conclusion drawn was that the Intel StrongARM SA-1100 was the most suitable processor of those that were investigated. This study helped to eliminate a large number of ARM processors from the search.

All of the processors listed in table 1-1, include on-chip cache memory, but allow it to be disabled via software, which will be necessary in an OBC design. Disabling the cache will translate to even lower power consumption, however, the computational performance will also decrease. A search for performance specifications of these processors with their caching features disabled proved unsuccessful.

It must be noted that the manufacturers' typical power consumption of the Intel and Cirrus Logic processors, listed above, are even further exaggerated as their test systems included the utilization of an LCD display.

As the chosen processor was not to be evaluated for the OBC of a specific satellite, most of the on-board peripherals that the processors included were not influential when the various ARM processors were compared.

2.3 Processor Details and Comparisons

The ARM7500FE from the Cirrus Logic (CL-PS7500FE) has the largest manufacturing process size of those listed above, at 0.5µm. Initially, this made it a possible choice, however, the processor is designed using relatively old 5-volt technology. This results in higher power consumption and a lower maximum clock speed. The processor also has a very basic memory controller. It only supports up to two static memory banks and does not fully support SRAM, which is necessary for an OBC design.

The Samsung KS32C50100 processor was designed as a network controller and its functionality would therefore be well suited to an OBC design. Its flexible memory controller, which boasts a full DMA controller, counted in its favour. Unfortunately, the relatively high power consumption and low performance of this processor did not make it a competitive candidate.

There are two ARM processor ranges from Intel: StrongARM and the new X-Scale. Both of these were designed to optimise low power consumption and high performance as they were designed for use in handheld devices such as Personal Diary Assistants (PDAs).

The Intel X-Scale PXA-250 had the most attractive MIPS per milliwatt ratio of all the processors. Unfortunately, the Intel X-Scale PXA-250 was ruled out, due to its 0.18μm manufacturing process. This very small process size would make the device extremely susceptible to effects of the space radiation environment.

This left only the StrongARM SA-1110, as it is the new revision of the SA-1100. The two processors are basically equivalent, except that the SA-1110 has improvements to its memory controller and other peripherals. The increased flexibility of the memory controller is highly beneficial to an OBC design, as it supports a wider range of memory types, such as variable latency I/O devices.

A major advantage of the SA-1100 is the fact that it has been flown successfully in satellite systems before (see section 3.8.3). The reliability of the SA-1110 should be inherently similar, as it is merely an upgraded version of the SA-1100.

2.4 Conclusion

According to the list of processor requirements, the Intel StrongARM SA-1110 was clearly the most suitable commercial ARM processor for use in an OBC design. This processor has a very impressive performance to power consumption ratio, and is manufactured using a relatively large process technology of 0.35μm.

The next chapter will discuss the details of the SA-1110 processor, which will be most important in a microsatellite OBC design.

CHAPTER 3

DETAILS OF THE SA-1110

The Intel StrongARM SA-1110 was identified as the most suitable ARM processor around which an OBC could be designed. Firstly, an overview of the processor will be given, followed by a detailed explanation of the features that make this processor well suited for an OBC design.

3.1 Overview of the StrongARM SA-1110

The StrongARM SA-1110 processor can be divided into the following four sections: Processing Core, Memory and PCMCIA Module, Peripheral Control Module and System Control Module. Figure 1-2 shows the block diagram of the SA-1110.

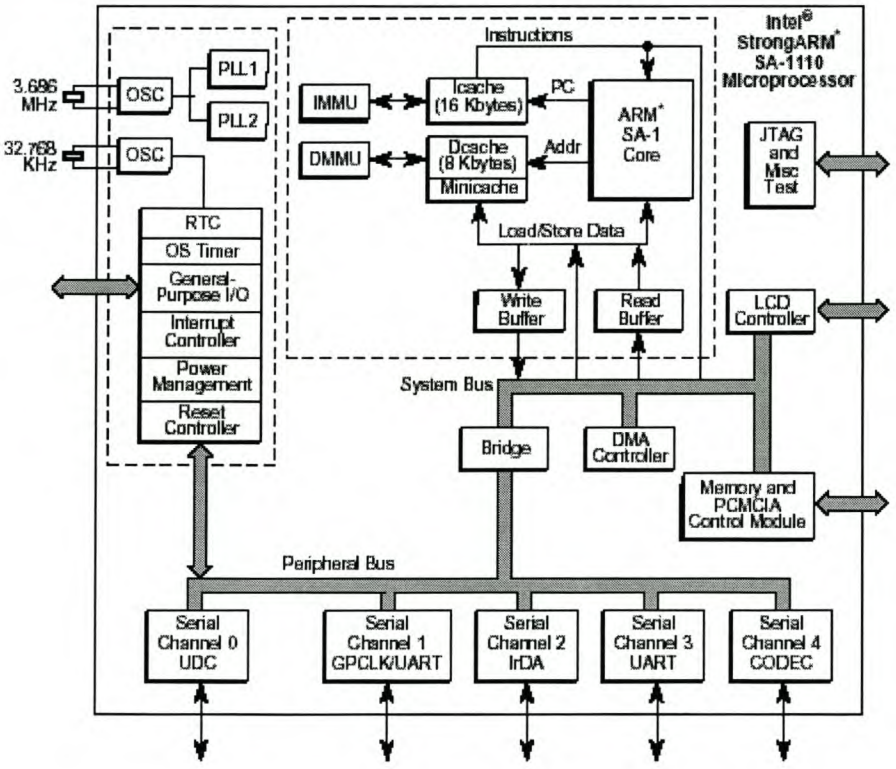

Figure 3-1: SA-1110 Block Diagram

3.1.1 Processing Core

The SA-1110 is Intel's third generation StrongARM processor, based on the ARM SA-1 core. It is software compatible with the ARM v4 (version four) architecture processor family. This

32-bit RISC microprocessor has a 16kB instruction cache and an 8kB write-back data cache. Each of these the cache memories have independent memory-management units (MMUs).

Stores are made using a four-line write buffer. The performance of specialized load routines is enhanced with the four-entry read buffer that can be used to pre-fetch data for use at a later time. A 16-entry (512 byte) mini-data cache is also available, and enhances caching performance when dealing with frequently used data structures.

3.1.2 Memory and PCMCIA Control Module

The memory controller supports a very wide range of memory types. The controller provides up to four banks for DRAM systems: fast-page-mode (FPM), extended-data-out (EDO), and/or synchronous DRAM (SDRAM).

Additionally, up to six static memory banks are supported. These static banks can control a combination of the following memory types: ROM, Flash (each with non-burst or burst read timings), Synchronous Mask ROM (SMROM), SRAM or variable latency I/O devices. The SA-1110 also provides a full PCMCIA (PC-Card) interface.

3.1.3 Peripheral Control Module

The peripheral control module (PCM) contains a six channel DMA controller, which is used to transfer data between memory and the on-board peripherals. Memory-to-memory DMA transfers are not supported.

The PCM also houses the following on-board peripherals:

- An LCD controller with support for passive or active displays
- Serial Port 0: A universal serial bus (USB) endpoint controller
- Serial Port 1: A combination general-purpose clock controller (GPCLK) and 16C550-type UART
- Serial Port 2: A serial controller with supporting 115 kbps and 4 Mbps IrDA protocols (can also be used as a 16C550-type UART)
- Serial Port 3: A general purpose 16C550-type UART, supporting up to 230kbps

- Serial Port 4: A combination multimedia communications port (MCP) and synchronous serial port (SSP).

3.1.4 System Control Module

The system control module (SCM) is also connected to the peripheral bus. It contains the following units used for general system functions:

- Two oscillator units, for the 32.768 kHz and 3.6864 MHz crystals
- A dual phase-lock loop circuit, to provide a primary 59-206 MHz core clock and a secondary 48 MHz clock for peripherals
- A real-time clock (RTC) with alarm
- An operating system timer (OST) for use as a general system timer as well as a watchdog timer
- Twenty-eight general-purpose I/Os (GPIO)
- An interrupt controller with up to 32 Fast Interrupt Requests (FIQs) and 32 Interrupt Requests (IRQs)
- A power-management controller that handles the transitions in and out of sleep and idle modes
- A reset controller that handles the various reset sources on the processor

3.2 Power Consumption

The low power consumption and complex power management system of the SA-1110 are key factors which motivate the use of this processor in an OBC design. The advantages of the low power consumption are two-fold. Not only does it conserve the limited power resources of the satellite, but it also minimizes the heat generated by the component. This is very important in the space environment, where heat dissipation is not possible by convection.

The power consumption of the processor is typically less than 350mW when running at its maximum core speed (206 MHz). This can be compared to, for example, a "mobile Pentium" processor, which delivers similar performance but uses nearly 8 watts of power [8].

3.2.1 Power Management

The power management logic of the SA-1110 controls the transition between three different modes of operation: run, idle, and sleep (see figure 3-2).

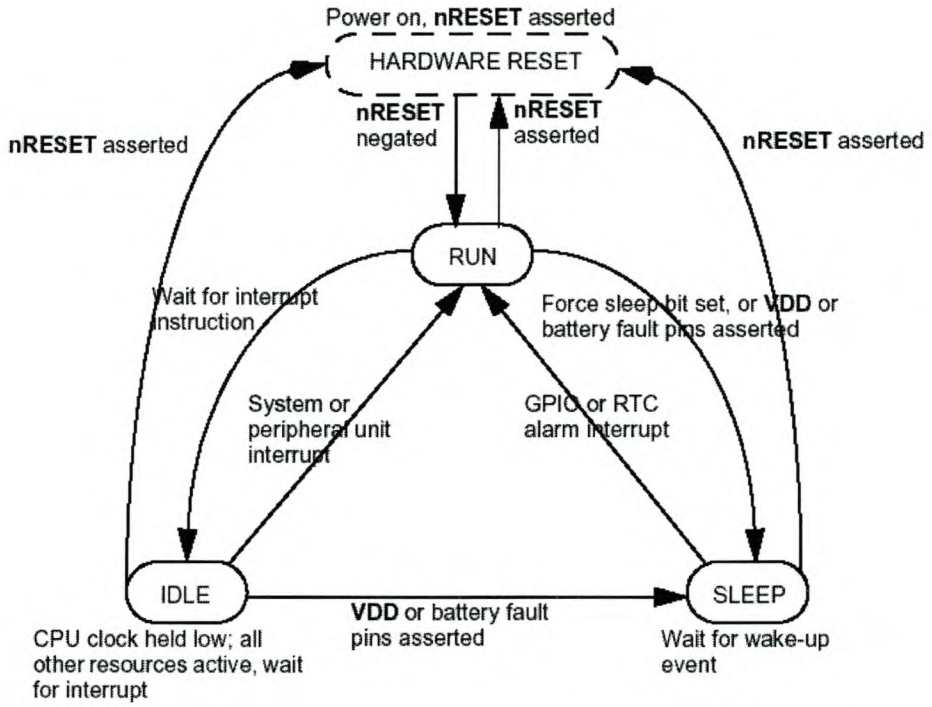

Figure 3-2: Run, Idle and Sleep Mode Transitions

These modes are used to reduce processor power consumption when certain functions of the processor are not required. The three operation modes are summarized in the sections below.

3.2.1.1 Run mode

This is the normal operating mode of the SA-1110. All power supplies are enabled, all clocks are running, and every on-chip resource is functional. This is the normal state of operation for the processor while it is executing code.

3.2.1.2 Idle Mode

In idle mode, the clock is stopped, but the rest of the processor modules remain operational. As the processor is static, the state information is preserved. Thus, on returning to run mode, the processor continues exactly where it left off. The typical idle mode power consumption is 100mW.

Idle mode can only be entered via software. It is meant to be activated by application software when the processor is not being used. Idle mode can be exited by the occurrence of any of interrupt (on or off chip).

This mode is a particularly useful feature of the SA-1110 power management system, as in this mode, it can provide a 70% power consumption reduction and code execution can be resumed quickly. This power saving method could be used hand-in-hand with a real-time operating system (RTOS). The SA-1110 could be forced into idle mode during the idle time of the operating system real-time period. Processing would then be resumed by the next interrupt-based timer tick, or an interrupt caused by an asynchronous event.

3.2.1.3 Sleep Mode

Sleep mode offers the greatest power savings and consequently, the lowest level of available functionality. When sleep mode is initiated, the processor shuts down much of the on-chip activity and applies an internal reset.

During sleep mode, the PWR_EN pin is driven low, indicating to the external system that the core power supply (VDD) can be driven to zero volts. With this supply disabled, most of the processor is switched off. This causes a dramatic power reduction, allowing the processor to consume as little as 165μW under ideal conditions.

Sleep mode is entered either via software or by asserting one of two input pins (VDD_fault and BATT_fault) that indicate a power supply fault. These external pins are intended to provide a "power low" and "supply out of regulation" signal to the processor, but they can be coupled to any external logic to force the processor into sleep mode when desired. This avoids the need for an external interrupt and software code to cause the processor to enter sleep mode at the request of an external source.

The I/O power supply (VDDX) remains active during sleep mode, as the processor needs to sample the wakeup sources: a real-time clock (RTC) alarm, or specified event (edge triggered interrupt) on a general purpose I/O (GPIO) input pin. The current drawn by the processor during sleep mode is highly dependent on the state of the GPIO pins. The sleep mode state machine actively samples all GPIO pins that are configured as inputs, even if they are not configured to cause a wakeup event. Special care must be taken to avoid unnecessary

switching of these inputs (especially at high frequencies) as this causes a significant increase in current.

The sleep mode state machine runs off the 32.768 kHz clock and a number of steps have to be carried out for a successful sleep shutdown sequence (see [2] for details of the sequence). With each step taking approximately 30µs, the shutdown sequences take approximately 90µs to complete. The wakeup sequence takes considerably longer (see [2] for details of the sequence). Firstly, the processor waits 10ms for the core power supply to stabilize, once it is re-enabled. The 3.6864 MHz oscillator can also be disabled during sleep mode, but this slows down the "wake-up" time, as the oscillator needs 150ms to stabilize. The wakeup time therefore varies from about 10ms to 160ms (as the 30µs steps of the state machines are insignificant).

Sleep mode cannot be used as freely as idle mode, due to its rather slow wakeup and shutdown sequences. It should only be used when relatively long periods of processor inactivity can be guaranteed.

3.2.2 Cache Memory Power Reduction

Another power saving aspect of the SA-1110 is that the on-chip cache memories and buffers can be disabled. The caches account for approximately 60% of the total die area of this processor [7]. Therefore, with the caches disabled in an OBC design, the power consumption should be significantly lower than the manufacturer's specifications.

3.3 The Memory Controller

The SA-1110 has a high-speed (103 MHz) memory bus and a flexible memory controller. It is able to provide an interface to a wide variety and large number of memory systems. This makes the processor very attractive for use in an OBC system.

The memory controller provides a static and dynamic memory interface. The dynamic memory interface supports up to four banks of DRAM, such as SDRAM and EDO DRAM. However, these types of memory are not suitable for satellite OBC systems as they are extremely susceptible to SEU [1]. The static memory interface is, however, highly suitable for a satellite OBC design.

The static memory interface has six chip selects (nCS[5:0]) and 26 bits of byte address (A[25:0]) for access of up to 64MB of memory in each of the six banks. This is more than enough for the memory system of a microsatellite OBC, as large mass memory systems aren't usually connected directly to the processors memory bus.

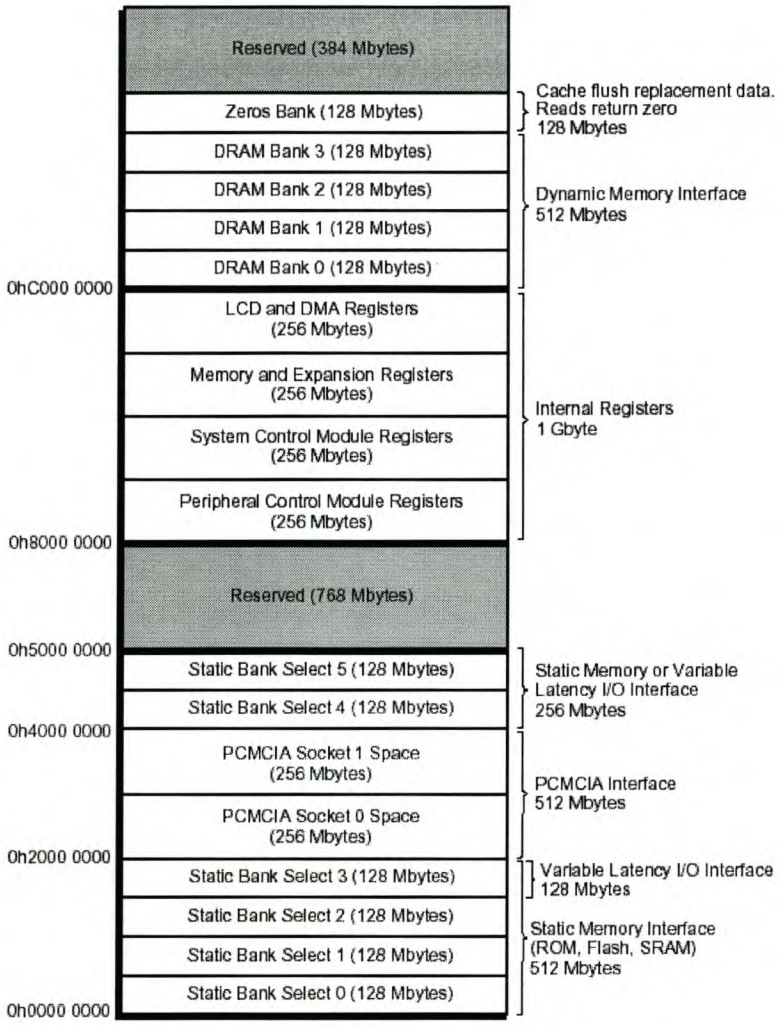

Figure 3-3: SA-1110 Memory Map

Each chip select is individually programmable for selecting one of the supported static memory types. The following static devices are supported, with 16 or 32-bit data bus widths:

- Non-Burst Mode ROM or Flash
- Burst Mode ROM or Flash (with only non-burst mode writes)
- SRAM
- Variable Latency I/O Devices

The six static memory banks should easily be sufficient for a typical microsatellite OBC memory system. Memory control signals to external memory mapped peripherals can be generated by external logic (such as an FPGA) to provide as many individual chip select lines as required.

The memory controller also includes a ready input signal, RDY, for use with SRAM-like variable latency I/O devices. Such devices can only be used on the upper three chip select lines (nCS[5:3]). The RDY pin (not present on the SA-1100) expands the flexibility of the memory controller dramatically. It is especially useful when designing memory mapped logic systems. This is applicable to a satellite OBC system, as the integration of an EDAC system can be simplified dramatically when a ready signal can be used. This is one of the most significant improvements of the SA-1110 over its predecessor.

3.4 System Timers

The SA-1110 provides two completely separate system timers, each driven by its own oscillator module.

3.4.1 Operating System Timer

The operating system timer is a 32-bit counter/timer that is clocked by the 3.6864 MHz oscillator. The operating system counter register (OSCR) is a free-running up-counter, which is not cleared by any reset. This module also contains four count match registers which can generate interrupts. The fourth match register can also be used as a watchdog match register, to implement a watchdog timer function. A match between this register and the OSCR will cause the SA-1110 to reset, rather than generate an interrupt, if the watchdog enable bit is set.

A watchdog timer is essential for any satellite OBC. Under normal circumstances, a watchdog is added to protect against the inevitable latent bugs in software. However, in the space radiation environment, program variables which reside in SRAM can become corrupted by single-event upsets (SEUs) and this causes unpredictable software behaviour. Thus, even though the running software is relatively bug-free, the corruption of data variables would eventually cause the software to crash. The subsequent reset caused by the watchdog would then reinitialise the system into a cleared, fresh state [1].

The watchdog timer function of the SA-1110 will prevent the need for an external watchdog timer in an OBC design.

3.4.2 Real-Time Clock

The real-time clock (RTC) provides a general-purpose real-time reference to the SA-1110. Normally, the real-time counter register (RCNR) is incremented on rising edges of the 1Hz clock. However, as the 32.768 kHz oscillator drives this clock circuitry, the counter can be incremented at a rate of up to 32.768 kHz (a timer with 30μs period). This makes the timer useful in generating a clock tick signal for a real-time operating system (RTOS) by using the RTC alarm interrupt (see [2] for more details).

Unfortunately, the RTC does not include a calendar function, which is very useful in an OBC, for diary functions. A calendar would have to be implemented via software.

3.5 General Purpose I/O

The StrongARM SA-1110 provides 28 general-purpose I/O (GPIO) port pins. Each of these pins may be configured via software as an input or output and as edge triggered interrupt sources (see [2], section 9-1). The general I/O port can therefore be configured to provide a large amount of external interrupt lines (up to 28). In many cases, this will help simplify the OBC design, as external interrupt controllers will not be needed.

This is very useful in an OBC environment, which typically has a large number of peripheral devices requiring external interrupt lines. A total of 15 external interrupt lines were needed on the SUNSAT OBC, where and external programmable interrupt controller (PIC) was needed [9].

3.6 On-Board Peripherals

The SA-1110 houses a large number of on-board peripherals. By using any of the on-board peripherals in a certain OBC design, the number of external peripheral components can be minimized. However, some of these peripherals are not suited for an OBC design, such as the LCD controller and multimedia communications port.

Fortunately, the StrongARM includes a Peripheral Pin Controller (PPC) which allows the pins used by the on-chip peripherals (except the USB controller) to be assigned as general purpose I/O pins. However, these pins cannot generate interrupts as the GPIO port pins can.

The use of the USB controller in an OBC system was only briefly considered, as the SA-1110's USB controller has a very bad reputation amongst embedded programmers. Several functions of the USB controller are reported to be unreliable. The USB controller does not incorporate a host controller and consequently can only act as a slave. This further limits its usefulness in an OBC design.

The on-chip UARTs (Serial Ports 2 and 3) are very basic and do not support modem control signals (such as RTS, CTS, DTR, and DSR). For this reason, the UARTs might not be ideally suited for use with the modems on a satellite. However, the GPIO port could be used to implement the necessary modem control lines via software control. The UARTs can be used for inter-subsystem communications on a satellite, such as a backup serial bus in a redundant system. These serial ports will also be very useful for debugging OBC software and could be included in an engineering model of an OBC.

The SA-1110's Serial Port 4 features a full-duplex Synchronous Serial Port (SSP). It may be configured to support the Motorola Serial Port Interface (SPI), National Microwire, Texas Instruments Synchronous Serial or a similar custom protocol. The SSP functions as a master only and can communicates to off-chip slave devices at a serial bit rate ranging from 7.2 kHz to 1.8432 MHz. The SSP could be very useful in a satellite OBC system, if one of the above mentioned serial interfaces were chosen for use in the design. The fact that the SSP can only act as a master may be a limiting factor when implementing a serial bus structure.

3.7 On-Chip Cache Memories and Read/Write Buffers

The cache memories and buffers reduce effective memory access time dramatically, and can improve code execution time significantly. Unfortunately, cache memories have proved problematic when used in the space radiation environment. The SA-1110 cache memories are disabled by default after a hardware or software generated reset.

As the cache memories and buffers can be enabled and disabled by software, this will allow experiments using these functions to be carried out easily. The StrongARM has an internal coprocessor (Coprocessor 15), which is used to enable and disable these features by writing to its registers. The SA-1110 User Manual [2] should be consulted for more details.

3.8 Reliability

The reliability and survivability of electronic components in the hostile environment encountered in space is extremely critical in any satellite system design. As the SA-1110 processor is the corner-stone of the entire on-board computer, all aspects concerning its reliability need careful attention.

The two major reliability concerns for a CMOS device (such as the SA-1110) in the space environment, are operating temperature and the effects of radiation.

3.8.1 Temperature Concerns

When LEO satellite systems are developed, industrial grade components are favoured, due to their superior operating temperature range (-40°C to 85°C). Unfortunately, the SA-1110 is only fabricated as a commercial processor and its recommended operating temperature range is 0°C to 70°C.

However, extensive environmental tests have been carried out on the Applied Data Systems (ADS) GCP2520 Graphics Client Plus System, which is an embedded system based on a StrongARM SA-1110 processor. The Computer Aided Life Cycle Engineering (CALCE) Electronic Products and Systems Centre carried out these tests and concluded that the StrongARM SA-1110 can operate successfully across the industrial temperature range [6].

3.8.2 Radiation Concerns

Unfortunately no data or reports are available regarding the radiation tolerance of the StrongARM SA-1110. The older, first generation, StrongARM SA-110 was tested by NASA Goddard Space Flight Centre (GSFC). The test board was a compact PCI card (CMA401) designed around the SA-110 by the Jet Propulsion Laboratory (JPL) [12].

The SA-110 was proton irradiated and only tested for single event effects (SEE), not total ionising dose (TID). The device was monitored for soft errors such as single event upsets (SEUs) and hard errors such as single event latch-up (SELs). The results showed only 2 events in approximately 20 krad (Si) of protons and no SEL was encountered.

The SA-110 was manufactured with an epitaxial layer, which gives some radiation protection. However, this CPU was manufactured by Digital Semiconductors who originally owned the rights to the StrongARM processor line. The current StrongARM range are manufactured by Intel and no information on the details of their manufacturing process could be obtained. For this reason we cannot assume that the Intel SA-1110 will share the same radiation resistant properties of the original Digital SA-110.

3.8.3 History of Use in Space

As mentioned before, a space history for the StrongARM SA-1110 could not be found. However, the SA-1100 does have a history of use in three satellite on-board computers. Fortunately, the SA-1110 is practically identical to the SA-1100 and both currently manufactured by Intel.

Surrey Space Technology Ltd (SSTL) used a SA-1100 processor in the design their OBC for the SNAP-1 nanosatellite. The SA-1100 processor was also used by the Radio Amateur Satellite Corporation (AMSAT). It was part of an experimental Integrated-Housekeeping-Unit (IHU), which is essentially an OBC, aboard their Phase 3D satellite. Finally, the United States Air Force Academy (USAFA) based the OBC of their FalconSat-2 (FS-2) nanosatellite on a SA-1100 processor. These three projects are discussed below.

3.8.3.1 The SNAP-1 Nanosatellite

The Surrey Nanosatellite Application Platform 1 (SNAP-1) is the United Kingdom's first ever nanosatellite. The COTS-based satellite was designed and built by Surrey Space Centre (SSC) and Surrey Satellite Technology Ltd (SSTL) staff.

The On-Board Computer (OBC) uses a 190 MHz StrongARM SA-1100 RISC processor (running nominally at 88 MHz). It includes 4M x 8-bit program memory, which is protected by a double-bit-correcting (16,8) code implemented in hardware. Communications interfaces

with the other satellite modules are based on the SA-1110's serial ports. One of the SA-1100's UARTs was configured for reception at 9600 bps while the second was configured for transmission at 38.4 kbps.

The machine vision system (MVS) on SNAP-1 is also designed around a StrongARM SA-1100. All image storage and processing is carried out by the SA-1100, which was equipped with 2MB of Flash-EPROM and 8MB of SRAM. Both of the StrongARM processor's had external CAN interfaces operated via the built-in SPI interface.

During SNAP-1's lifetime of two years, there was no reported failure of the SA-1100 processors.

3.8.3.2 AMSAT Phase 3D

The on-board computers of previous AMSAT satellites (OSCAR-10 and OSCAR-13) were based on the 8-bit COSMAC 1802 processor. As these processors were soon to be obsolete, a secondary experimental on-board computer was added to the Phase 3D satellite. This secondary OBC (or IHU) was to be tested, in flight, to determine whether it could be used as a possible primary OBC in future AMSAT satellites.

The OBC system included 128k x 32-bit of hardware protected fast SRAM memory (EDAC) and 8 megabytes (2M x 32-bit) of software protected SRAM memory.

Although current information about the IHU-2's flight history is minimal, AMSAT have reported that the IHU-2 is operating successfully and reliably. In should also be noted that their processor is running with its cache memory enabled.

3.8.3.2 The FalconSat-2 Nanosatellite

FalconSat-2 (FS-2) is a nanosatellite developed by United States Air Force Academy (USAFA). It is based largely on core modules from the SNAP-1 nanosatellite. The COTS-based StrongARM OBC is one of the modules that have been adopted, due to its previous success in SNAP-1. The satellite is due for launch in January 2003.

CHAPTER 4

A TYPICAL OBC DESIGN

In order to perform an accurate evaluation of the StrongARM SA-1110 for use in an OBC system, it was necessary to develop a typical OBC system based on this processor. Usually the functional specifications for an on-board computer are determined by the satellite system requirements, but in this case the design was not intended for a specific satellite project.

For this reason, the OBC's local peripheral interfaces and inter-subsystem interfaces, were simply chosen as examples that could be used for a future satellite design. Several of these choices were based upon other satellite OBC designs, which have proven successful.

4.1 OBC Design Overview

The OBC is a complex custom computer system, based on the SA-1110 processor. A basic block diagram of the entire OBC is shown in figure 4.1. It consists of the following:

- *Processor*
  - Intel StrongARM SA-1110
- *Memory System*
  - Boot ROM (Flash)
  - Program ROM (Flash)
  - Program RAM (EDAC Protected SRAM)
- *On-Board Peripherals*
  - Real-Time Clock (RTC)
  - Analog-to-Digital Converter (ADC) for Current Sensing
  - Temperature Sensor
  - Dual Channel Serial Communications Controller (SCC)
  - LVDS Communications Channel
  - RS-232 Line Drivers (for SA-1110 UARTs)

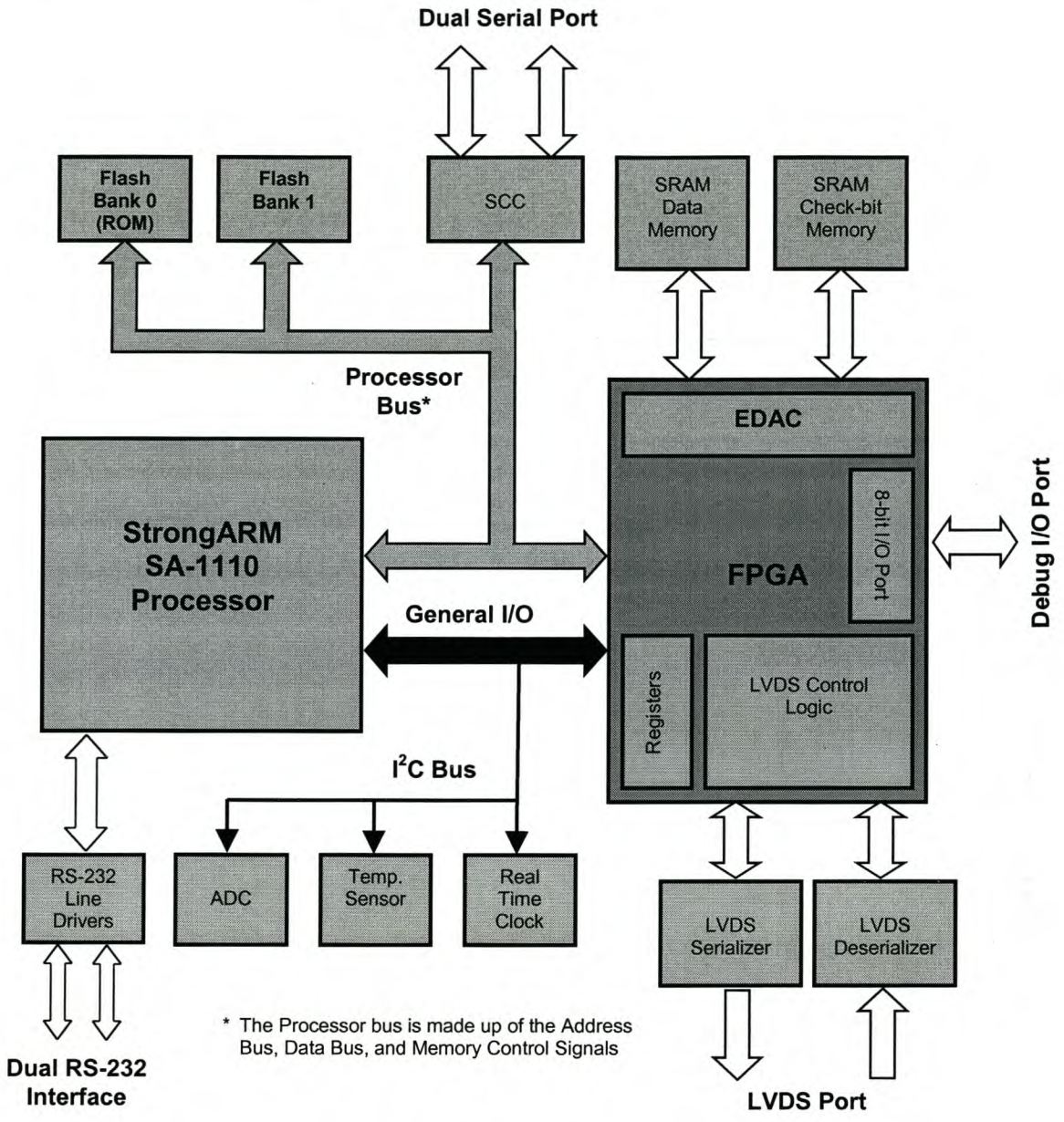

Figure 4-1: OBC Block Diagram

The OBC also includes an Altera EP1K30 FPGA, which implements the following modules:

- Hardware Registers
- EDAC system
- LVDS control logic
- All the other necessary glue logic

The OBC provides the following interfaces for communications with the satellite's other subsystems:

- I^2C Bus
- Two SCC channels (typically used for a modem interface)
- Two UARTs (SA-1110 on-chip peripherals)
- High-Speed Full-Duplex LVDS Channel

All of the OBC peripherals and interfaces, as well as the system support circuitry, will be discussed in the appropriate sections that follow. Full schematic diagrams for the OBC design are included on the CD-ROM provided. The physical layout of the OBC is shown in Appendix A.

4.2 The StrongARM Processor

Intel suppies a range of StrongARM SA-1110 processors, with two available maximum core clock speed options: 133 MHz or 206 MHz. The functionality of these two processors is identical. As expected, the lower speed option exhibits lower power consumption and can be used in a design where its computational performance is sufficient. The higher speed option (part no. GDS1110BD) was used in this study.

As discussed in the previous chapter, the StrongARM SA-1110 processor is a highly integrated device with many on-board peripherals and functions. The on-board peripherals that required hardware-level integration are discussed below.

4.2.1 Serial Ports

To provide a simple interface between a personal computer (PC) and the StrongARM processor, a set of RS-232 transceivers were connected to the on-board UART2 and UART3 of the processor. These 16C550-type UARTs do not include modem signals, and therefore level conversions are only required for the data transmit and data receive lines.

A single Maxim MAX3222E device was sufficient as it provides two receivers and two drivers. These devices would not be used on an actual OBC, but were included to facilitate the development of the OBC software.

4.2.2 General Purpose I/O (GPIO) Port

The general I/O port provides external interrupt lines to many of the devices on the OBC. These will be mentioned in the appropriate sections that follow.

Six of the GPI0 lines (15,16, 21, 22, 26 and 27) of the processor are connected to the FPGA, to allow simple control signals to pass between them, or allow the FPGA to interrupt the processor.

A summary of the GPIO port configuration is given in table 4-1, below.

General Purpose I/O Port	Purpose
GPIO 0	Temp. Sensor Alarm Interrupt
GPIO 1	SCC Interrupt
GPIO 2	SCC Interrupt Acknowledge
GPIO 3	I^2C SCL (Clock)
GPIO 4	I^2C SDA (Data)
GPIO 5	Debug LED 1
GPIO 6	Debug LED 2
GPIO 7	Debug LED 3
GPIO 8	Debug LED 4
GPIO 9	Debug DIP Switch 1
GPIO 10	Debug DIP Switch 2
GPIO 11	Debug DIP Switch 3
GPIO 12	Debug DIP Switch 4
GPIO 13	RTC Alarm Interrupt 1
GPIO 14	RTC Alarm Interrupt 1
GPIO 15	FPGA Debug Port Pin 1
GPIO 16	FPGA Debug Port Pin 2
GPIO 21	FPGA General I/O
GPIO 22	FPGA General I/O
GPIO 26	FPGA General I/O
GPIO 27	FPGA General I/O (Reset)

Table 4-1: GPIO Configuration

4.3 The Power Supply System

The power supply system for the OBC design was rather complex, as the devices operated from a variety of power supply voltages. The four power supplies, and the components that they deliver power to, are listed below:

- **1.8V** for the StrongARM processor core
- **2.5V** for the FPGA's core
- **5.0V** for the SCC and the associated level-shifters
- **3.3V** for all the other devices, including the FPGA and Processor I/O interfaces

The layout of the power supply system is as follows:

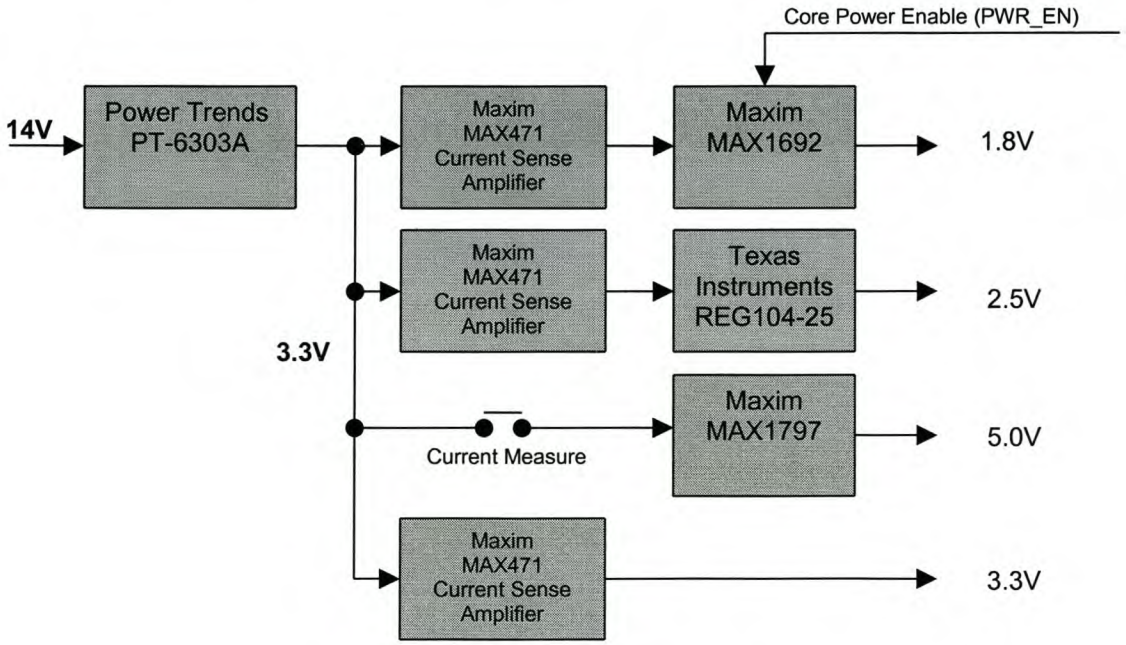

Figure 4-2: The OBC Power Supply

A 1.8V power supply providing an enable input, was used to complement the SA-1110 core power supply enable line (PWR_EN). By using this output to enable the 1.8V supply, the SA-1110 will enforce the required power supply sequencing by holding PWR_EN de-asserted until the 3.3V supply is sufficiently high. This also allows power conservation to be maximized during sleep mode, when PWR_EN is de-asserted, disabling the 1.8V supply. Current sense amplifiers (MAX471 devices) were inserted directly after each supply, except for the 5.0V supply. These current sense amplifiers will allow the system software to monitor

the current drawn from the 1.8V, 2.5V and 3.3V supplies, by using an ADC. The current sense amplifiers were biased to provide a one-volt-per-amp output signal to the ADC. Details of the ADC will be discussed in section 4.8.5.

4.4 The Reset Circuit

The following reset circuit, based on a MAX6315, was used to provide a reliable reset signal to the SA-1110. The MAX6315 device acts as a 3.3V power supply monitor and also provides de-bouncing for a push button switch. The device guarantees minimum reset time of 150ms required by the processor.

Additionally, the SA-1110 is also held in the reset state until the FPGA is configured. The CONF_DONE output of the FPGA was used to do this.

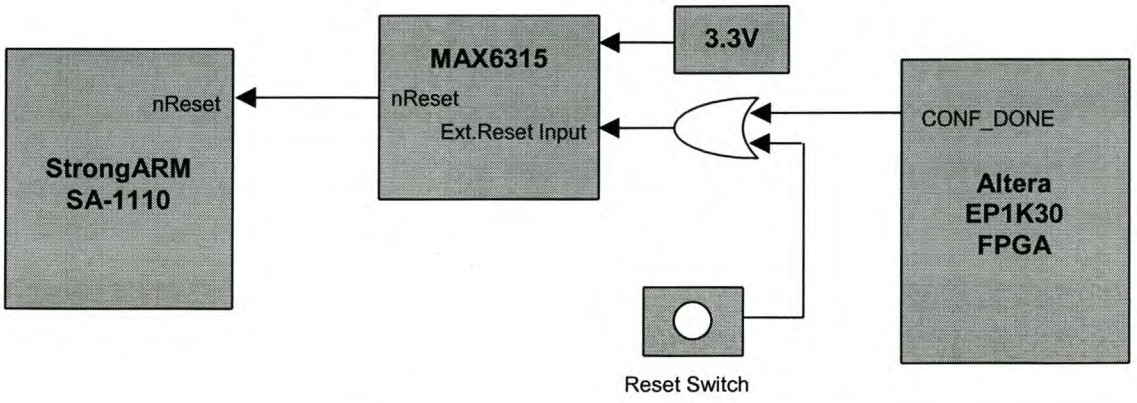

Figure 4-3: SA-1110 Reset Circuit

Many of the logic circuits in the FPGA will require a reset signal after the FPGA has been configured, to ensure that they are correctly initialised. One of the general I/O lines (GPIO 27) between the processor and FPGA was used for this purpose. The processor can therefore reset the FPGA logic when desired.

4.5 The Memory System

The basic structures of the memory system for microsatellite OBCs do not vary to a large extent. A memory system common to most microsatellite OBCs was used in this design. It

consists of: Non-volatile Boot Code Memory, Non-volatile Program Code Memory, and volatile Program Code/Data Memory.

An FPGA also forms part of the memory system, as it is connected to the processor bus. This is required for the FPGA-based EDAC system. Additionally, this allows peripherals connected to the FPGA (such as the LVDS) to be accessed as memory mapped devices, via registers. These aspects will be discussed later, in the appropriate sections.

A block diagram of the memory system, showing the mapping of the processor's chip select lines is shown in figure 4-4, below.

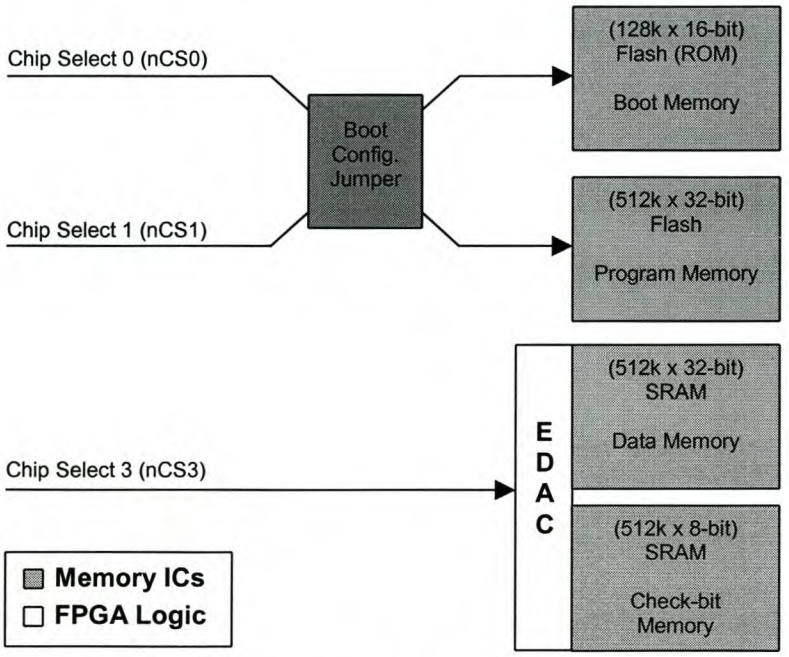

Figure 4-4: OBC Memory Configuration

The DRAM memory controller is not used in the design. Only the StrongARM's static memory banks are using to support the OBC memory system.

4.5.1 Boot Memory (ROM)

The first memory bank of the StrongARM (chip select 0) starts at memory address zero. This bank must be occupied by non-volatile bootable memory, as code fetching starts here after a reset. The boot loader code will reside here.

On a true satellite OBC, this memory would be a one-time programmable ROM device (OTPROM), as the integrity of the boot code is highly critical. However, to aid the initial boot software design, removable (PLCC package) flash memory was used. Two Am29LV001BB (128k x 8-bit) boot sector Flash memory devices provide a 16-bit wide boot block. These were the smallest 3.3V PLCC flash devices available. This memory will be referred to as the ROM in this design.

4.5.2 Program Memory (Flash)

The second bank of memory, associated with nCS1, is occupied by 2MB of flash memory. Two Intel TE28F800-B3 Advanced Boot Block Flash Memory devices make up the 512k x 32-bit memory space. Normally, this bank would be used to run program code from, as the SA-1110 can access it via the full 32-bit data bus.

When flash memory devices are used on a satellite, dual-supply devices, without charge pumps, are favoured. This is due to the fact that the charge pumps, which provide the higher programming voltage, are highly susceptible to radiation. The internal controllers found on most modern flash devices are also an area of concern, as their state can become corrupted during SEU [2].

The flash devices chosen for this design are dual-supply. This allows reliability improving circuitry to be incorporated into this evaluation OBC system, as would be the case on a satellite. The devices do however, include charge pumps, so that only a 3.3V external programming voltage (V_{PP}) is required.

The programming voltage should only be available to the device when absolutely necessary. This helps prevent unintentional writes to the memory, which could be caused by the controller state corruption (mentioned above). The data protection circuit shown in figure 4-5 was implemented [5]. It allows the programming power supply to be disconnected from the flash device. The nLockFlash signal is connected to the FPGA and can be set via the FPGA-based hardware setup register, SETUP_REG, discussed in section 4.6.

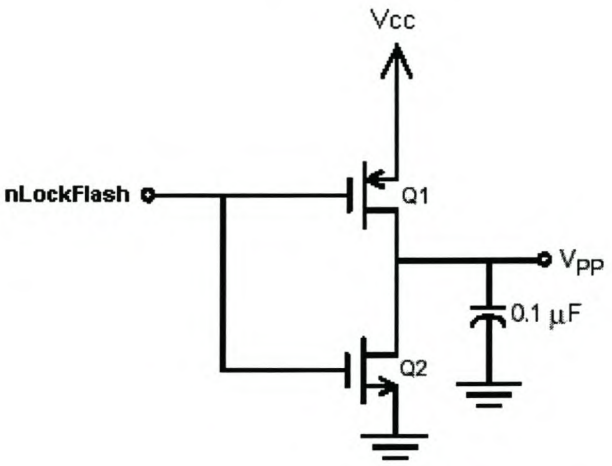

Figure 4-5: Flash Data Protection Circuitry

The flash devices offer two hardware lockable boot sectors of 8kB each, providing a total of 32kB between the two. This area is ideal for storing critical data, or boot code. The write protection input (#WP) associated with these sectors is connected to the FPGA, and can also be set via the hardware setup register.

4.5.3 Boot Configuration Jumpers

A set of jumpers was included in the memory system design, to allow the boot code location to be changed between the ROM and flash devices. By default, the ROM will be mapped to chip select 0, and flash to chip select 1, as on most OBCs. To take advantage of the boot sector flash, the jumpers can be set as in figure 4-4(b).

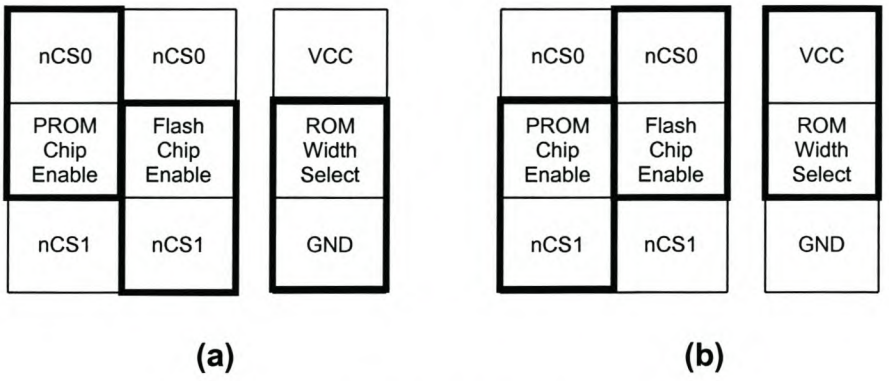

Figure 4-6: Boot Memory Configuration

Additionally, the SA-1110's ROM size select (ROM_SEL) pin needs to be driven to indicate the bus width of the boot memory. A high on this pin indicates a 32-bit device (Flash), while a low indicates a 16-bit device (ROM).

4.5.4 Protected Program Data/Code Memory (SRAM)

The third memory bank on the OBC consists of RAM, as required by the SA-1110. Unfortunately, the very high density of modern RAM devices have resulted in reduced memory cell sizes, making them more prone to SEU induced errors. Even the memory systems of satellites, operating in LEO orbits are at risk, especially when passing through the South Atlantic Anomaly (SAA) [1]. For this reason, it was necessary to implement and error correction and detection (EDAC) system for this memory bank.

The 2MB of EDAC protected SRAM are based on five Toshiba TC55V8512J-15 8-bit devices. They are arranged as 512k x 32-bit for data, and 512k x 8-bit for the check-bits. In order to take advantage of the StrongARM's high-speed memory bus, high-speed SRAM devices were used, with an access time of 15ns. This was important for the overall system performance, as the EDAC itself would introduce additional memory access delays.

This protected memory is accessed using chip select 3. This allows the SA-1110 to address the memory as a variable latency device, by using the ready input. This allowed for a faster and more flexible EDAC design, the details of which will be discussed in the following section.

4.5.5 Error Detection and Correction (EDAC) Unit

There are a large number of EDAC systems that have been used on satellite systems. These can generally be divided into the following categories:

- Software based EDAC
- A Commercial EDAC Integrated Circuit (IC)
- FPGA based EDAC

The disadvantages of using a completely software based EDAC system are the highest. This technique introduces extra processing overhead and the system software has to be designed to support this feature explicitly.

Commercial EDAC ICs (such as the IDT49C465) provide a simple off-the-shelf EDAC solution. However, the high prices and high minimum order quantity of these devices, made them unacceptable for this project.

The final option was to implement EDAC logic in an FPGA. An Altera EP1K30QC208-1 device was used. The highest speed grade was chosen, to minimize the delays associated with the EDAC logic.

Two possible methods of designing the EDAC system were identified: using a Triple Modular Redundancy (TMR) design, or a design using a data-coding scheme. A TMR system was used on the AMSAT P-3D IH-2 (discussed previously). Unfortunately, this technique requires triple the number of memory devices and was therefore rejected.

EDAC based on a data-coding scheme was to be used. There are a large number of coding schemes available, but Hamming code based designs are simple and are generally considered to be acceptable for use in an OBC [1]. The particulars of Hamming coding scheme used are described below.

4.5.5.1 The Hamming Code

For each data-word written to memory, a coded pattern or check-bit word, is generated. The new word (the data-word plus the check-bit word) can be termed a valid code. The Hamming code establishes a distance-of-4 between one valid code and another. This means that to go from one valid code to another, 4 bits have to change [13].

It can be shown that a distance-of-4 code enables you to detect all single and double-bit errors and correct all single-bit errors. To implement a distance-of-4 code for the SA-1110's 32-bit data bus, a 7-bit check-bit word is required.

4.5.5.2 The EDAC VHDL Design

The (39,32) Hamming code can be easily implemented using VHDL code. The logic which implements the (39,32) Hamming code is based on an open-source VHDL module provided by the European Space Agency (ESA). The basic structure of the check-bit generation (figure 4-7) and error correction system (figure 4-8) is shown below.

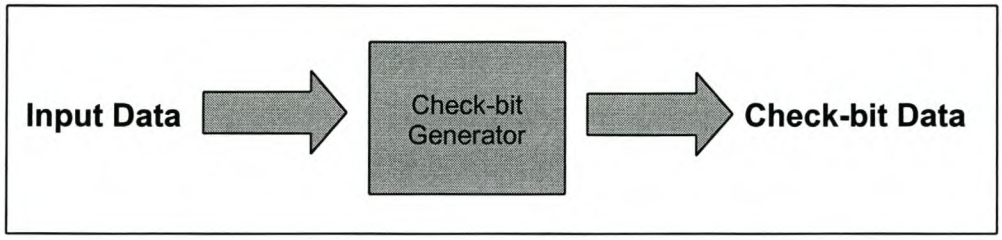

Figure 4-7: Check-bit Generation

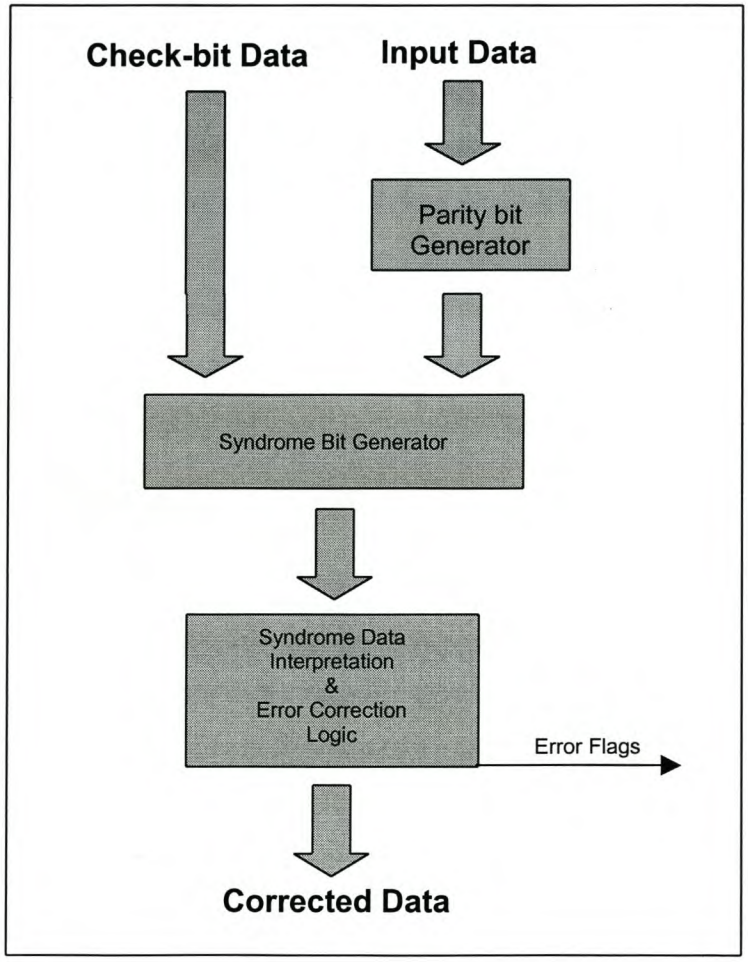

Figure 4-8: Error Correction System

For further detail of the Hamming code logic implementation, the VHDL source code can be referenced on the CD-ROM provided.

A flow-through architecture was used to implement the EDAC system, as can be seen in figure 4-8. In contrast to a bus-watch architecture, a flow-through EDAC improves the throughput of the EDAC operation and simplifies the interface between the CPU system bus and the memory bus.

The address signals generated by the processor, pass directly to the SRAM devices, while the data and control signals pass through the EDAC system.

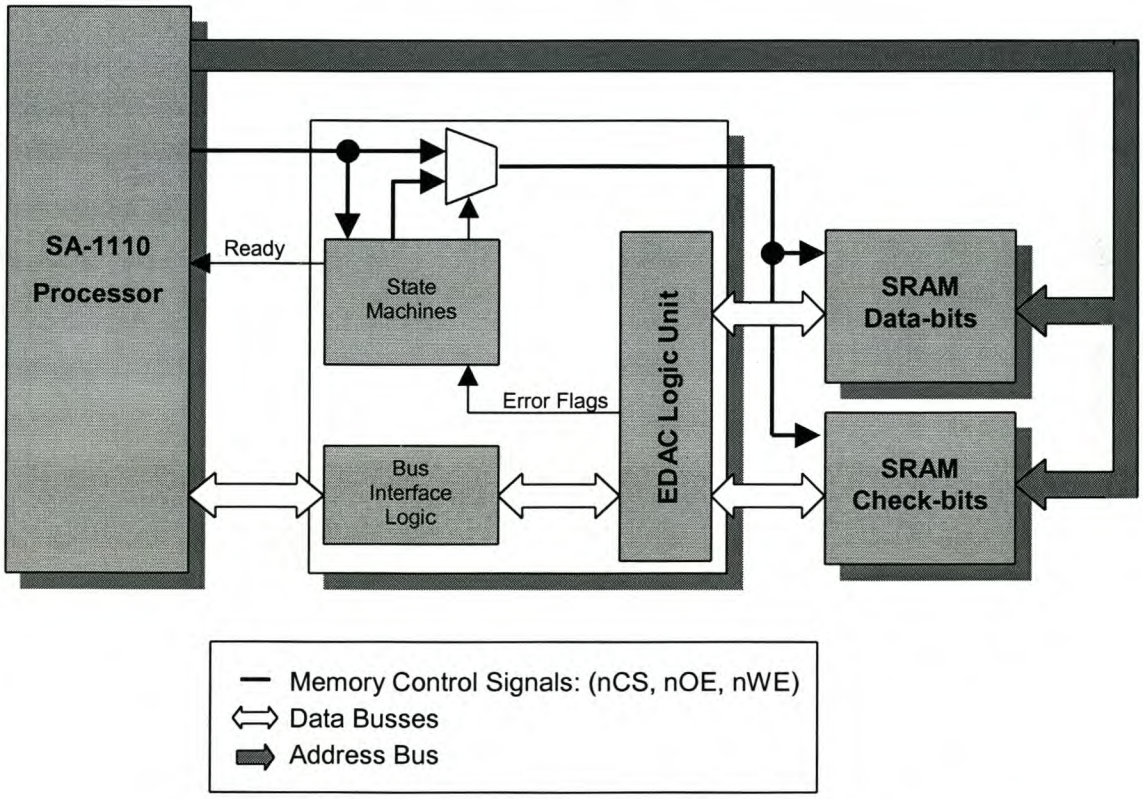

Figure 4-9: Block Diagram of FPGA Based EDAC System

This EDAC system interprets the memory control signals produced by the processor to enable either a read-cycle or write-cycle state machine. A 40 MHz clock, derived from the external 80 MHz oscillator of the FPGA, drives these state machines. The details of the EDAC read and write cycles are discussed below.

Read-Cycle

During the first part of a read operation, the processor memory control signals are merely allowed to pass directly to the SRAM devices. For this period, the ready input to the processor is not activated. If there is no error, the ready input is enabled. This informs the ARM that it is may latch the requested data in and complete its read cycle.

If the EDAC logic detects an error, the state machine enters an extended read cycle, in order to write the corrected data and check-bits back to the SRAM devices. During this period, the state machine takes full control of the SRAM control signals (nCE, nWE and nOE). Once the write to SRAM is complete, the ready signal is enabled, completing the extended read cycle.

Write-Cycle

The write-cycle state machine allows the memory control signals from the processor to be passed through to the SRAM devices. The state machine asserts the ready input of the processor, once enough time has elapsed for the EDAC logic to generate the check-bits. At this time, the memory control signals (still generated by the processor) latch the data into the SRAM devices, and the write-cycle is terminated.

Many EDAC systems use a periodic scrubbing method, to prevent the accumulation of single bit errors [1]. This is normally a software process that has to be scheduled at regular intervals. This technique can be ineffective when the rate of SEUs is much higher than the rate at which the memory is being "scrubbed".

This EDAC system was designed to write the corrected data back to the memory, within the same active memory cycle, via an extended-read-cycle (as explained above). This also makes the EDAC completely transparent to the processor.

4.6 FPGA-based Registers

The FPGA also implements a set of 32-bit registers for the OBC system. A hardware setup register (SETUP_REG), LVDS transmit data register (LVDS_TXD) and LVDS receive data register (LVDS_RXD) are included.

SETUP_REG controls the protection circuitry of the flash memory, enables or disables the LVDS system, and can be used to access an 8-bit I/0 port on the FPGA. This I/O port was incorporated mainly for debugging purposes. The data registers associated with the LVDS communications channel will be discussed in section 4.7.2.

Register Name	Physical Byte Address
SETUP_REG	0h1000 0000
LVDS_TXD	0h1000 0004
LVDS_RXD	0h1000 0008

Table 4-2: FPGA-based Register Locations

Bits	10	9	8	7..0
Data	LVDS_off	nFlashWP	nFlashLock	I/O Port

Table 4-3: SETUP_REG Bit Fields

The FPGA logic interprets the memory control signals of the ARM processor in a similar way to the EDAC system, however, the address bus has to be read too. These registers are mapped to nCS2.

4.7 The LVDS Communications Channel

An LVDS (Low Voltage Differential Signalling) system was included to provide a high-speed, full-duplex serial communication channel to other satellite subsystems. LVDS is best used in continuous streaming data transfers and could therefore be used to access a mass memory device, for instance.

LVDS provides a means of sending data along a twisted pair cable at high speed, with low power and with excellent EMC performance. These features make LVDS ideal for satellite on-board data-handling applications [14].

The LVDS channel is based on the DS92LV1021 Serializer and DS92LV1212 Deserializer from National Semiconductors. Most of the data and control lines of the LVDS devices are connected directly to the FPGA, to make the design as flexible as possible. As the FPGA

provides all of the necessary control logic for these devices, the amount of additional support circuitry is minimized.

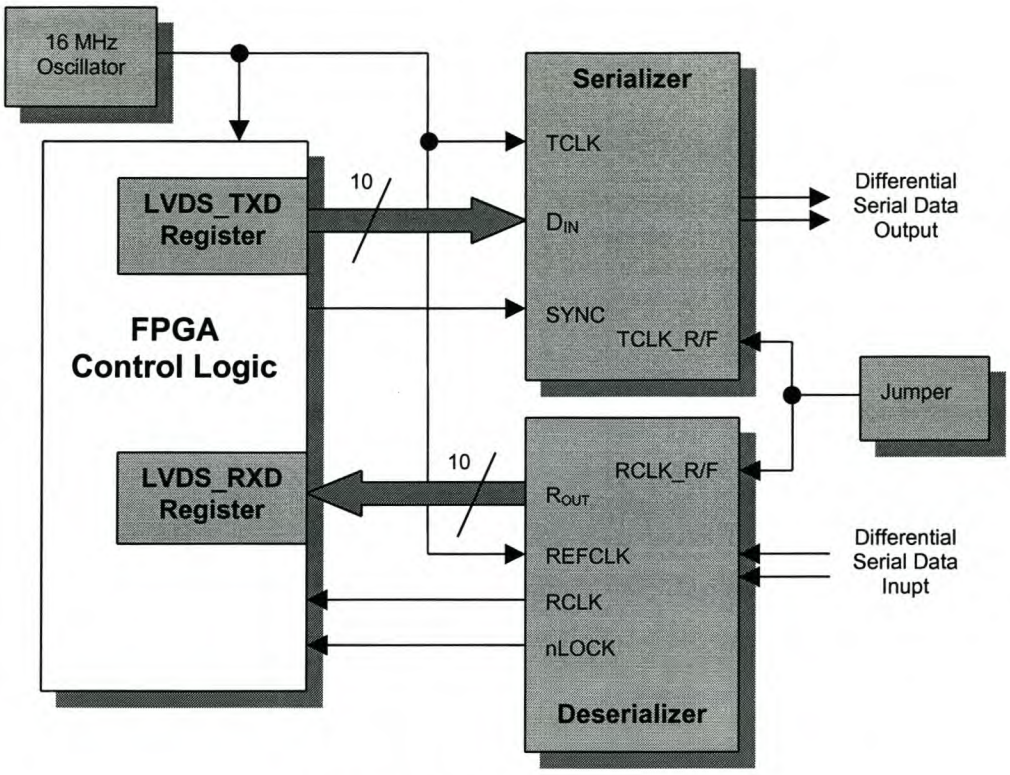

Figure 4-10: LVDS System

The LVDS system includes a 16 MHz oscillator, which drives the FPGA-based control logic and the clock inputs required by the serializer and deserializer. As the system is based on a 16MHz clock, it provides a 160 Mbps data-link.

4.7.1 The Serializer and Deserializer Basic Operation

The DS92LV1021 transforms a 10-bit wide parallel CMOS/TTL data bus into a single high speed Bus LVDS serial data stream with embedded clock. Data at D_{IN} is clocked into the serializer by the TCLK input. The edge of TCLK used to strobe in data is selectable via the TCLK_R/F pin (high for rising edge and low for falling edge). This is selectable via a jumper on the board. The serial outputs (DO±) can be tri-stated by the DEN input being driven low.

The DS92LV1210 receives the Bus LVDS serial data stream and transforms it back into a 10-bit wide parallel data bus and separates the clock. This recovered clock signal (RCLK) is used to strobe the parallel data at R_{OUT}. The RCLK_R/F pin is used to select the active edge for

strobing of R_{OUT} data. It is also connected to the jumper mentioned above. The outputs ROUT0–ROUT9, LOCK and RCLK can be tri-stated by driving the REN input low.

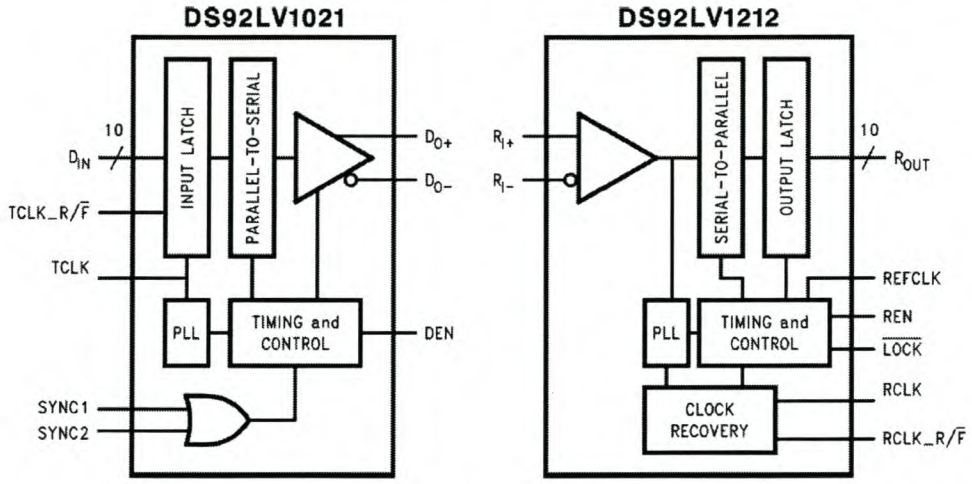

Figure 4-11: LVDS Serializer and Deserializer

As the LVDS system implements a point-to-point connection, only the receiver end of the transmission media needs to be terminated by a 24-100Ω resistor.

4.7.2 The LVDS Control Logic

To correctly initialise the two LVDS devices after power-up, correct cycling of the power down (nPWR_DN) pins is required. This was accomplished using a state machine, housed in the FPGA. The state machine is initiated after the LVDS reset signal is toggled.

After the power-up sequence, the next step requires that the deserializer and serializer be phase locked to each other. Phase locking is accomplished by the serializer sending SYNC patterns to the deserializer. The serializer sends SYNC patterns whenever its SYNC1 or SYNC2 inputs are held high.

The nLOCK output of the deserializer is high whenever the deserializer is not locked. To simplify the synchronization between the two devices, the nLOCK output was connected to the SYNC inputs via the FPGA. This closed-loop system guarantees that enough SYNC patterns are sent to achieve deserializer lock.

The data on the deserializer output (R_{OUT}) is valid while the nLOCK output is low. If the serializer loses lock (nLOCK goes high) during data transmission, up to 5 cycles of data that was previously received can be invalid. These 5 cycles should be re-transmitted, which means that a communications protocol would have to be implemented for reliable data transfers.

The FPGA control logic keeps the outputs of LVDS devices enabled, by driving the DEN and REN signals high, as these are the only devices on the LVDS bus. The devices can be forced into a power-down mode by setting bit 10 (LVDS_off) in the SETUP_REG register discussed previously.

The SA-1110 processor can access the LVDS channel via the memory mapped LVDS_TXD and LVDS_RXD registers, housed in the FPGA. The LVDS_TXD register is used to load a value to the D_{IN} pins of the serializer. The LVDS_RXD register contains the value of the R_{OUT} pins while nLOCK is low. The register is zero when nLOCK is high.

The current VHDL code only provides a simple way to test the LVDS hardware by indirectly writing to D_{IN} and reading from R_{OUT}. However, the hardware design is such that it can be expanded considerably. A hardware level communications protocol including error correction as well as transmit and receive buffers could be added to make the LVDS system more useful.

4.8 I^2C Serial Bus

An example of a low-speed serial bus was included in the OBC design. In this design, the bus was used to support a telemetry system for the OBC. A system based on an Inter-Integrated Circuit (I^2C) bus was chosen due to its simplicity and the wide range of devices that support it. The extremely low current consumption and high noise immunity of the I^2C bus also makes it very attractive for satellite applications. An I^2C internal bus has also been used on Stanford University's CubeSat.

4.8.1 I^2C Background

The bus physically consists of 2 active wires called SDA (data) and SCL (clock) and a ground connection. Both of these lines use 2.2kΩ pull-up resistors, as defined by the I^2C specification. Both SDA and SCL are bi-directional.

Earlier versions of the I^2C specification only supported Standard-mode (up to 100 kbps) and Fast-mode (up to 400 kbps) devices. However, the current specification, version 2.1 [15], includes new High-speed mode which allows data rates of up to 3.4 Mbps. High-speed mode devices are fully downward compatible with Fast- or Standard-mode (F/S-mode) devices for bi-directional communication in a mixed-speed bus system. The current specification also supports up to 10-bit addressing, allowing the use of up to 1024 devices.

4.8.2 Implementing an I^2C bus

Two of the SA-1110's general purpose I/O pins (GPIO 3 and 4) were used to connect the processor to the I^2C bus. To use the processor on the I^2C bus, a software driver had to be written (see section 5.3.2.2). Unfortunately, the StrongARM's GPIO pins can only support a standard-mode I^2C controller, due to the slow rate at which the pins can be toggled.

The OBC design also makes provision for the FPGA to be attached to the I^2C bus via a set of jumpers. However, a VHDL core implementing an I^2C controller was not written. An FPGA-based I^2C controller would allow a Fast-mode or High-speed mode bus to be used. The processor would also be able to access the I^2C bus via the FPGA by extending the FPGA-based register logic, discussed in section 4.6.

The SA-1110's software controller or an FPGA-based controller avoid the need for an additional external I^2C controller device to be added to the system.

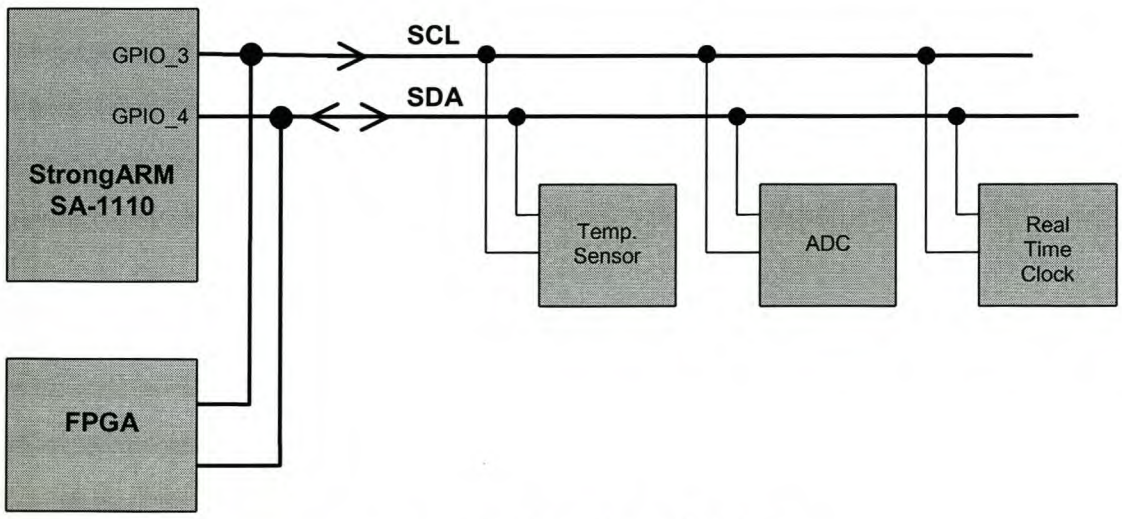

Figure 4-12: OBC I^2C Bus

Even though the I^2C bus supports multiple masters, the SA-1110 would be configured as the sole I^2C master, with all of the I^2C peripheral devices acting as slaves. The functions of the slave devices are discussed below.

4.8.3 Temperature Sensor

The Maxim temperature senor (MAX6654) used in this design acts as precise digital thermometer. It reports the temperature of both a remote P-N junction and its own die (local temperature). The local temperature gives an accurate indication of the printed circuit board (PCB) temperature while a freestanding 2N3906 PNP transistor was used for the remote sensor.

The support circuitry for the device is very simple and no calibration is needed. The MAX6654 device also provides an interrupt line to the SA-1110 on its GPIO Port (bit 0). The interrupt is triggered if the local or remote temperatures are outside a user-programmed range.

4.8.4 Real-Time Clock

A Dallas DS1337 Real-time clock (RTC) was added to the OBC to provide an accurate time reference to the system. The RTC also includes a calendar which can be very useful to the diary function of OBC [1]. This prevents the need for a software-based clock/calendar being developed, using the SA-1110's basic on-board RTC.

The DS1337 also contains two time-of-day alarms, which can generate interrupts according to the user specified alarm times. These interrupt lines are connected to the SA-1110's GPIO Port (bit 13 and 14). These alarms can therefore be used to schedule events on the OBC.

4.8.5 Analog-to-Digital Converter (ADC)

An ADC was included to measure the outputs of current sense amplifiers. An Analog Devices AD7828 8 channel 12-bit ADC was used. The device requires very little support circuitry, making its implementation simple.

The first three single ended channels (CH0, CH1, CH2) of the ADC are connected to the current sense amplifiers of the 3.3V, 2.5V and 1.8V supplies.

4.9 Serial Communications Controller (SCC)

Generally, the SCC will provide an interface to the communications subsystem of a satellite, which consists of various modems. It can also be used for medium-speed inter-subsystem communications. A Zilog Z85C30 was chosen as it has been used successfully in satellite programs, such as the University of Utah's CATSAT.

The SCC has two independent full-duplex serial channels. The SCC supports most standard synchronous and asynchronous serial transfer protocols, and has built-in cyclic redundancy code (CRC) and High-level Data Link Control (HDLC) capabilities.

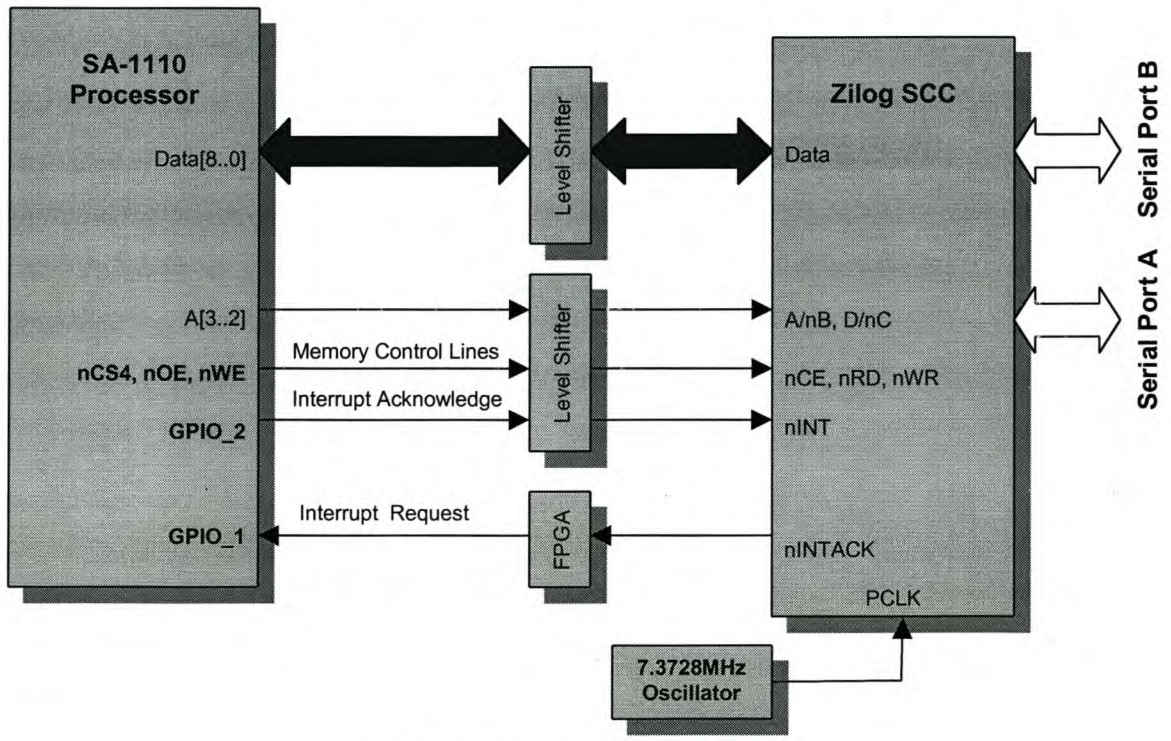

Figure 4-13: SCC Circuit

All the registers of the SCC can be accessed via two address lines. One address line selects the channel (A/nB) and another selects the command or data (D/nC) register.

Unfortunately the Zilog SCC is a 5-volt device, and the StrongARM's pins are not 5-volt tolerant. For this reason, two 8-bit tri-statable level-shifters (Fairchild 74LVXC4245) had to be included. The interrupt request line of the SCC uses two of the 5-volt tolerant pins of the FPGA to connect to the processor.

CHAPTER 5

SOFTWARE DEVELOPMENT

In order to complete an evaluation of the StrongARM-based OBC design, sufficient software had to be written. Before the processor could be programmed, it was necessary to investigate the ARM architecture programming model.

The next step was to choose and configure a suitable programming environment for the StrongARM OBC. The final section of this chapter covers the software that was written to start-up and test the OBC system.

5.1 The ARM Programmers Model

This section provides a summary of the ARM v4 Programmers Model. For more detail, the ARM Architecture Reference Manual [10] should be consulted.

The ARM architecture incorporates these typical RISC features:

- A large uniform register file.
- A load/store architecture, where data-processing operations only use registers, not memory.
- Simple addressing modes, load/store addresses determined from register contents and instruction fields.
- Uniform and fixed-length instruction fields, which simplify instruction decoding.

In addition, the ARM architecture provides:

- Control over Arithmetic Logic Unit (ALU) and shifter in all data-processing instructions.
- Auto-increment and auto-decrement addressing modes to optimise program loops.
- Load/store multiple instructions to maximize data throughput.

- Conditional execution on all instructions to maximize execution throughput.

5.1.1 ARM Processor Modes

The ARM architecture supports seven processor modes, shown in table 5-1, below.

Processor Mode	Abbreviation	Description
User	usr	Normal Program Execution
FIQ	fiq	Support high-speed data transfers
IRQ	irq	Used for normal interrupt handling
Supervisor	sve	A protected mode for operating systems
Abort	abt	Implements virtual memory and memory protection
Undefined	und	Supports software emulation of hardware coprocessors
System	sys	Runs privileged operating system tasks

Table 5-1: ARM v4 Processor Modes

Programs are usually run in User mode, where some system resources are not accessible. An exception is required to change from User mode to another mode.

The other processor modes are privileged modes, which have full access to system resources and may change between modes freely. Five of these are exception modes: FIQ, IRQ, Supervisor, Abort and Undefined. These are entered when a specific exception occurs. They each have access to additional registers to avoid corrupting the User mode state when the exception occurs.

The other privileged mode, System mode, is not entered via an exception and has the same basic registers available as in User mode. It is not subject to the restrictions associated with User mode.

5.1.2 ARM Registers

The ARM has 31 general-purpose registers and six status registers, shown in table 5-2. All of these registers are 32 bits wide.

Modes						
	Privileged Modes					
		Exception Modes				
User	**System**	**Supervisor**	**Abort**	**Undefined**	**Interrupt**	**Fast Intr**
R0	R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7	R7
R8	R8	R8	R8	R8	R8	R8_fiq
R9	R9	R9	R9	R9	R9	R9_fiq
R10	R10	R10	R10	R10	R10	R10_fiq
R11	R11	R11	R11	R11	R11	R11_fiq
R12	R12	R12	R12	R12	R12	R12_fiq
R13	R13	R13_svc	R13_abt	R13_und	R13_irq	R13_fiq
R14	R14	R14_svc	R14_abt	R14_und	R14_irq	R14_fiq
PC	PC	PC	PC	PC	PC	PC
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
		SPSR_svc	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

Shows that original register has been replaced by alternative banked register, specific to the exception mode

Table 5-2: ARM Register Organization

There is a different register bank for each processor mode. At any time, 15 general-purpose registers, the program counter and one or two status registers can be seen.

5.1.2.1 General Purpose Registers

Registers R0-R7 are un-banked registers, which are available in all processor modes. These registers do not have any special uses implied by the architecture.

Registers R8-R14 are banked registers, where the physical register referred to depends on the current processor mode. Registers R8-R12 are general-purpose registers which have two banked registers each. One set is used for all processor modes other than FIQ mode, and the other set is used for FIQ mode. Register 13 holds the stack pointer (SP). Each exception mode has its own (banked) stack pointer. Each exception mode also has its own (banked) Register 14, known as the Link Register (LR). It is used to hold a subroutine or exception return address, which is copied back to the program counter when the subroutine or exception ends.

Register 15 holds the Program Counter (PC), which is common to all processor modes.

5.1.2.2 Status Registers

The current program status register (CPSR) is accessible from all processor modes. It contains the Arithmetic and Logic Unit (ALU) status flags, interrupt disable bits, the current processor mode, and other status and control information. Each exception mode has a saved program status register (SPSR) that is used to preserve the value of the CPSR when the exception occurs. The format of the CPSR and SPSR is shown in figure 5-3.

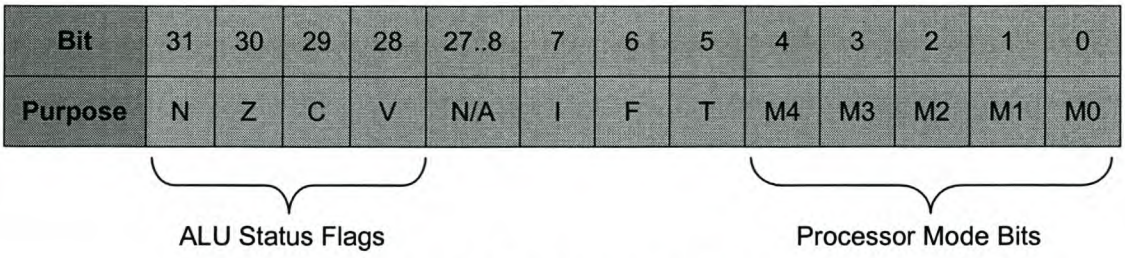

Figure 5-1: Status Register format

The ALU (or condition) status flags are tested by most instructions, to determine whether the instruction is to be executed. The I and F bits, are set to disable interrupts or fast interrupts, respectively. The T bit should always be zero for the SA-1110. The five mode bits determine the mode in which the processor is operating.

5.1.3 Exceptions

The table 5-4 shows the seven different exceptions supported by the ARM architecture, and the processor mode in which the exception is processed.

FIQ mode provides more private registers (R8-R12) than IRQ mode. This prevents the need for those registers to be saved (on the stack) and thereby minimizes the overhead of the context switching. The FIQ vector is also deliberately last, so the exception handler can be placed directly at the vector address, without requiring a branch instruction. This too improves the FIQ response time.

Exception Type	Mode	Address
Reset	Supervisor	0x0000 0000
Undefined Instructions	Undefined	0x0000 0004
Software Interrupt (SWI)	Supervisor	0x0000 0008
Prefetch Abort	Abort	0x0000 000C
Data Abort	Abort	0x0000 0010
IRQ (Normal Interrupt)	IRQ	0x0000 0018
FIQ (Fast Interrupt)	FIQ	0x0000 001C

Table 5-3: Exception Vectors

5.1.4 The ARM Instruction Set

The ARM v4 instruction set is made up of 45 highly integrated instructions. The details of these instructions are too extensive to be included in this report. They are provided in the ARM Architecture Reference Manual [10].

5.2 The Programming Environment

In order to write software for the OBC, it was necessary to set up a programming environment for the ARM processor. The ARM-ELF GNU Cross Compiler (GCC) tool chain was used due to its popularity amongst ARM programmers and most importantly, it is available free of charge. Fortunately, such a tool chain had already been set up for the ARM v4 architecture, on a local Linux server, by Francois Retief.

5.2.1 The Linker-Script

A custom linker-script had to be written according to the memory configuration of the OBC, given in figure 5.1. The linker-script defines a *.text* section for the executable code which is mapped to the ROM. It also defines a *.data* section for write-able initialised data and a *.bss* section for the write-able non-initialised data. These sections are mapped to the Protected SRAM.

The linker script also specifies the entry point, which indicates the first instruction of the boot code. The corresponding entry point in the assembler code will be mapped to the start of the *.text* section.

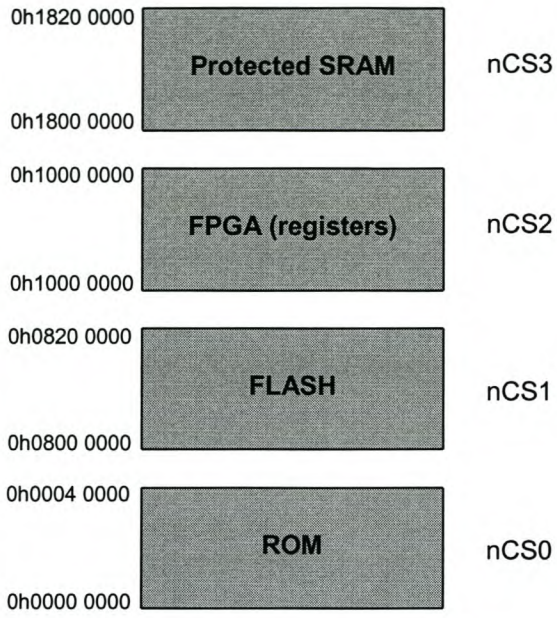

Figure 5-2: OBC Memory Map

5.3 OBC Software

Developing software for a complete satellite OBC is an enormous task as the software is generally application specific. However, in order to evaluate the StrongARM processor for use in an OBC, it is only necessary to test the basic functionality of the OBC system. The software has to configure and test each feature of the hardware system, so that comments can be made on the success of the design.

The software written for the OBC can be separated into the following categories:

- Boot code
- System Drivers
- System Utilities
- Test Programs

5.3.1 Boot Code

It is the responsibility of the boot code to correctly initialise the SA-1110 processor after a reset occurs. The boot code consists of two initialisation stages:

1. Initialising the execution environment (memory, exception vectors, stack, etc). This is carried out in assembler code.
2. Initialising the application (C variables). This is carried out in C code.

These two stages are discussed in detail in the sections that follow.

5.3.1.1 Initialising the execution environment

By default, after a reset, little-endian mode is enabled, and all caches and buffers are disabled. The processor enters 32-bit supervisor mode and sets the I and F bits in the CPSR (disabling the interrupts). The processor then jumps to the reset vector, at address 0h0000 0000. The assembler code located here completes the following tasks:

- Sets up the exception vectors.
- Initialises the memory system.
- Initialises the stack pointer registers.
- Sets the Processor Speed.
- Setup and Enable interrupts.
- Puts the processor in User mode.

These tasks are discussed in more detail below.

Exception Vector Setup

The exception vectors are organized according to table 5-3 and made up of branch instructions to the handler of each exception.

Memory System Initialisation

Code fetching starts from the ROM selected by static chip select 0 (nCS0). After a reset, the static bank 0 is automatically configured for the slowest non-burst ROM/Flash and the ROM_SEL pin determines the bus size of the boot ROM.

The SA-1110 registers, MSC2, MSC1, and MSC0, are read/write registers containing control bits for configuring static memory corresponding to chip select pairs nCS[5:4], nCS[3:2], and nCS[1:0], respectively. The control data for each static bank is therefore 16 bits long. This includes three timing fields: ROM Delay First access (RDF); ROM Delay Next access (RDN); ROM/SRAM Recovery time (RRR) and two type fields: ROM Type (RT) and ROM Bus Width (RBW) fields. The layout of these registers is given in the SA-1110 User Manual [ref], page 141.

The initialisation code sets up these control registers for optimum use of the OBC memory system, according to the memory specifications in table 5-5.

Memory	Chip Select Line	Bus Width	Access Time
ROM	nCS0	16-bit	90ns
Flash	nCS1	32-bit	90ns
FPGA Registers	nCS2	32-bit	N/A
Protected SRAM	nCS3	32-bit	~120ns (variable)

Table 5-4: OBC Memory Specifications

Stack Pointer Initialisation

Each processor mode has its own stack pointer. The stack pointers for User and Supervisor mode must be initialised, along with the stack pointers for every exception mode that is to be used. This basic boot loader only supports IRQ interrupts. Generally, Abort and Undefined modes are not used in a simple embedded system.

The stack memory starts at the top of the SRAM, at address 0x1820 0000. Firstly, 256 bytes are allocated to the Supervisor mode, followed by 256 bytes to IRQ mode. This is followed by the User mode stack.

Setting the Processor Speed

Following a reset, the processor core clock is set to its lowest frequency of 59 MHz. The core clock frequency is configured by writing to the core clock configuration field (CCF[4:0]) in the power manager phase-locked loop (PLL) configuration register

(PPCR). By default, the boot code sets the processor's core clock to its maximum speed of 206 MHz.

Setup and Enable Interrupts

The IRQ interrupts are enabled by clearing the I bit in the CPSR. The boot code also masks all of the interrupts by writing a zero to the Interrupt Controller Mask Register (ICMR). All interrupts are also configured as IRQs, as opposed to FIQs, by writing to the Interrupt Level Control Register (ILCR).

Enter User Mode

The final step of the environment initialisation causes the processor to enter User Mode.

5.3.1.2 Application Initialisation

After the environment has been initialised, the boot code sequence continues with the application initialisation by entering the C code. This is basically an initialisation of the variables and memory space required by the C code.

The initial values for any initialised variables must be copied from ROM to RAM (their run-time location). All other non-initialised variables must be initialised to zero. The code responsible for this is executed immediately after the branch to the C code. The code takes advantage of the symbols defined by the linker script.

At this point, the system is ready to run the intended application.

5.3.2 Software Drivers

In order to make use of some of the system peripherals, software drivers had to be written. These drivers are used to configure the peripherals and provide a simple interface to the programmer.

5.3.2.1 Serial Port Driver

A serial port driver was written for the OBC to allow software to communicate with a PC terminal. This provides a simple means for status information to be displayed by test programs. This would prove invaluable during software debugging.

The driver configures the on-board UART3 of the StrongARM processor by simply writing to the appropriate serial port control registers. The registers are programmed to set up the UART for a baud rate of 57.6 kbps, and a serial data frame of 8 data bits, no parity bits and one stop bit (8,N,1). The receive and transmit operation of the UART, as well as the receive interrupt, are enabled. For details of the SA-1110's UART control registers, see the SA-1110 Developers Manual [2].

The serial port driver unmasks the SA-1110's UART 3 interrupt, so that interrupts can be generated for the UART's receive operations. To process the interrupts, an interrupt handler had to be written. As all of the SA-1110's interrupts are configured as IRQs, code was written to identify the source of the IRQ interrupts, by reading the interrupt pending register (ICPR).

If bit 17 is set in the interrupt pending register, the UART 3 interrupt service routine is called. The UART 3 status register (UART3_SR0) identifies the actual interrupt source. When a receive occurs, the interrupt handler merely reads the UART 3 data register (UART3_DR) and clears the receive interrupt flag (UART3_SR0, bit 3).

The serial port driver provides a simple function to transmit strings of bytes. It polls the UART's transmit FIFO status bit (UART3_SR0, bit 0) and writes a byte to the UART 3 data register when appropriate.

5.3.2.2 I^2C Port Driver

A software driver was written to allow the StrongARM processor to access the I^2C peripherals via two GPIO lines. The driver only allows the SA-1110 to act as the sole bus master. The following basic functions implement the various bus events required for a simple I^2C controller:

- Start Condition
- Stop Condition

- Get Acknowledgement
- Give Acknowledgement
- Give Negative Acknowledgement
- Transmit Byte
- Receive Byte

Using these functions, all the transactions required for communicating with slave devices can be generated.

5.3.3 Software Utilities

To aid software development for the OBC system, several system utility programs were written. All these utilities use the StrongARM's UART 3, via a RS-232 port to connect to a PC. Any simple terminal program can be used on the PC to access these utility programs. The serial port of the PC should be configured for a baud rate of 57.6 kbps and 8-bit data, one stop-bit and no parity.

The system utility software has the following features:

- Upload Program Code to Flash
- Flash Memory Dump
- SRAM Memory Dump
- SA-1110 Special Registers Dump

5.3.4 Test Programs

In order to test all of the StrongARM's functions, a large number of small programs had to be written. The program code is available on the accompanying CD-ROM. A list of the test programs are given below, however, the details of the software will not be covered in this document.

- Watchdog Setup and Test
- Real-Time Clock Setup and Test
- Sleep Mode Test

- I^2C Slave Device Tests (RTC, Temp Sensor, ADC)
- On-Chip UART Test

CHAPTER 6

TESTS AND MEASUREMENTS

Many test programs were written for the SA-1110, to test the functionality and performance of the OBC system and the SA-1110 processor. The results of these tests are discussed in the sections that follow.

6.1 Power Consumption

The power consumption of any satellite sub-system is very important. It was therefore necessary to measure the amount of power consumed by this OBC system. Unfortunately, the power used by the OBC is influenced by many varying factors: the frequency at which the memory and peripherals are accessed, and the specific code that is running. It is also difficult to measure or calculate the power consumption of each component, as they share common bus interfaces and power supplies.

6.1.1 Running Mode

In order to obtain realistic power consumption results, software was written to use the main functions of the OBC continuously. The hardware and software conditions under which the tests were carried out are specified below:

- 206 MHz Core Clock Speed
- Program code executed from the 32-bit Flash
- EDAC System Enabled
- An SRAM read and write every 1ms
- LVDS in loop-back mode
- I^2C Devices are being accessed at a rate of 1 kHz

The current sense amplifiers (discussed in section 4.2) allow the currents which feed all of the 3.3V devices and the two secondary power supplies (2.5V and 1.8V) to be measured. The total power consumption of the OBC can be calculated using these measurements, and measuring the 5.0V current, via the PCB's inline test points.

	Average Current (mA)	Power Consumption (mW)
5.0V Device Supply	19	95
3.3V Device Supply	122	403
2.5V FPGA Core	41	103
1.8V SA-1110 Core	62	112
	TOTAL	**718**

Table 6-1: OBC Power Consumption Details

The total power consumption of the OBC was calculated as 718mW. This value can be compared to the total power delivered at the 14V main power input, where the operating efficiency of the 3.3V power supply must be considered. The 14-volt supply delivers 1152mW, but the 3.3V step-down converter is operating at an efficiency of approximately 60%. The effective power supplied is therefore 691mW, which is very similar to the initial calculation.

6.1.2 Idle Mode

Enabling idle mode was not as straightforward as expected, as it required operations on the co-processor 15 registers, which were not investigated in this project. A successful transition to idle mode was therefore not achieved.

6.1.3 Sleep Mode

When the SA-1110 was forced into sleep mode, the power consumption of the OBC dropped to 511mW. In this state, the power consumption of the SA-1110 is very low (theoretically 165uW) and can be ignored.

The SRAM and Flash memories, are in stand-by mode, and will only consume a small amount of power. The LVDS devices are also powered down before sleep mode is entered, by writing to bit 10 of the SETUP_REG register.

6.1.4 The SA-1110 Core's Power Consumption

The current used by the SA-1110's core, in running mode, could be measured by the current sense amplifier situated at the 3.3V point feeding the step-down switching regulator. When making the power consumption calculations it was important to note that the 1.8V regulator was operating at an efficiency of about 90% across the measured load current range. The graph below shows the power consumption of the processor's core at its 11 possible operating frequencies.

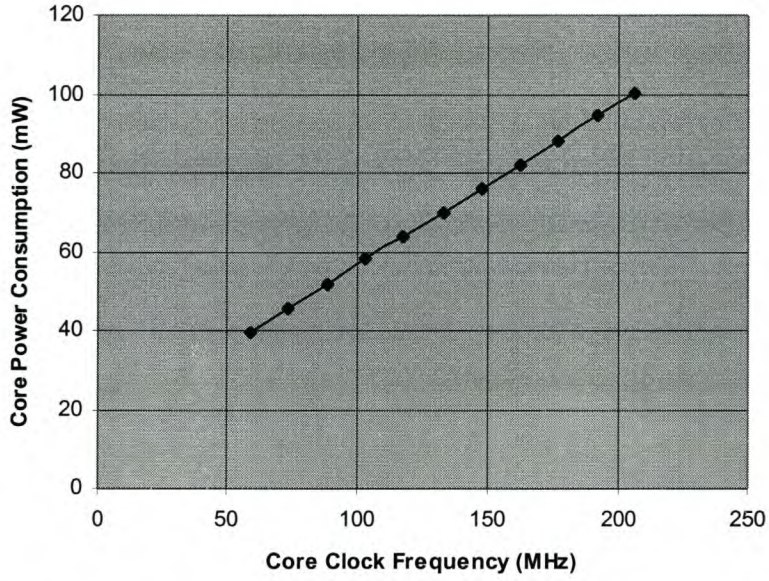

Figure 6-1: SA-1110 Core Power VS Frequency

6.2 Memory System Performance

The memory system is usually the main performance bottleneck of a computer system. For this reason, it was important to measure the access time of the protected memory system that had been developed.

High speed SRAM, with an access time of 15ns was used in the design. However, it is the FPGA-based EDAC system and the StrongARM's memory controller that determine this memory bank's overall performance.

The StrongARM's memory controller imposes non-ideal timing constraints on its memory banks for variable latency I/O devices. For example, there are long delays between the nCS and nOE/nWE signals, which are not programmable. The memory controller also introduces

delays associated with the sampling of the Ready input. The theoretical minimum memory cycle time for the 15ns SRAM, using a variable latency memory bank (ready signal always asserted), was calculated as 58.3ns. Access to non-variable latency static banks is much more efficient, but do not allow the ready input to be used.

The time delay associated with the FPGA implementation of the EDAC logic is approximately 17ns to generate the check-bits and 27ns to perform a complete error correction. Each step in the read and write cycle state-machines takes 25ns in the current EDAC implementation. These are the main factors influencing the performance of the EDAC system.

The average memory cycles times of the protected memory system are summarized in table 6-1, below.

Memory Cycle	Cycle Time (nCS = low)
Write Cycle	~88ns
Read Cycle	~124ns
Extended Read Cycle	~172ns

Table 6-2: Protected SRAM Memory Performance

It must be noted that the memory cycle times, represented above, may vary by up to 25ns as the 40 MHz clock which drives the FPGA-based EDAC system is not synchronized to the StrongARM's 106 MHz memory clock.

The logic analyser screen shots below confirm the memory performance (as in table 6-2) and show the working of the EDAC system.

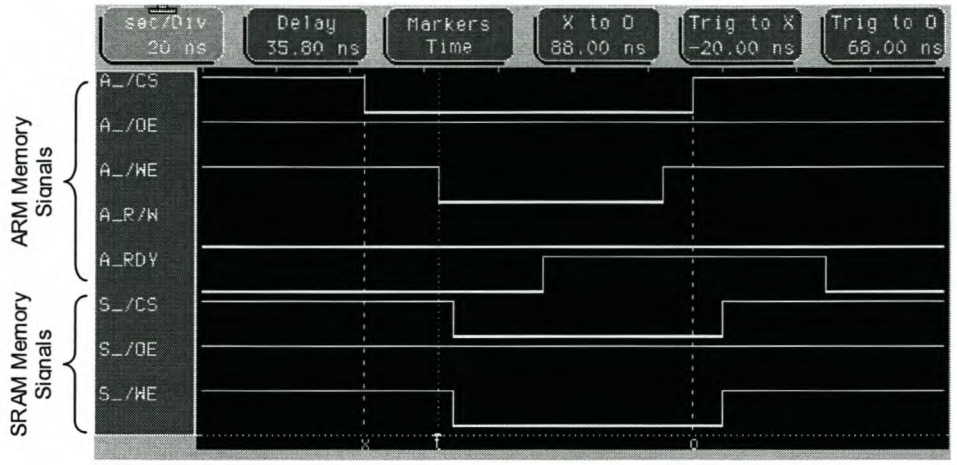

Figure 6-2: EDAC Write Cycle

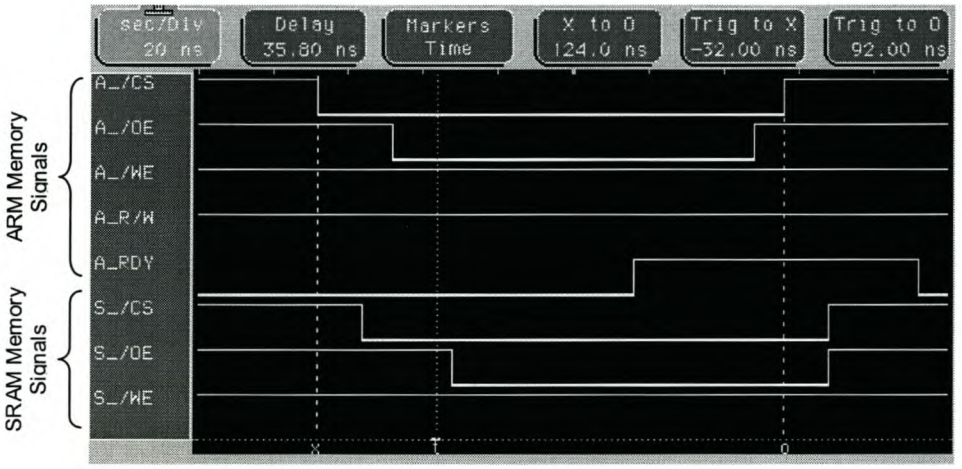

Figure 6-3: EDAC Read Cycle

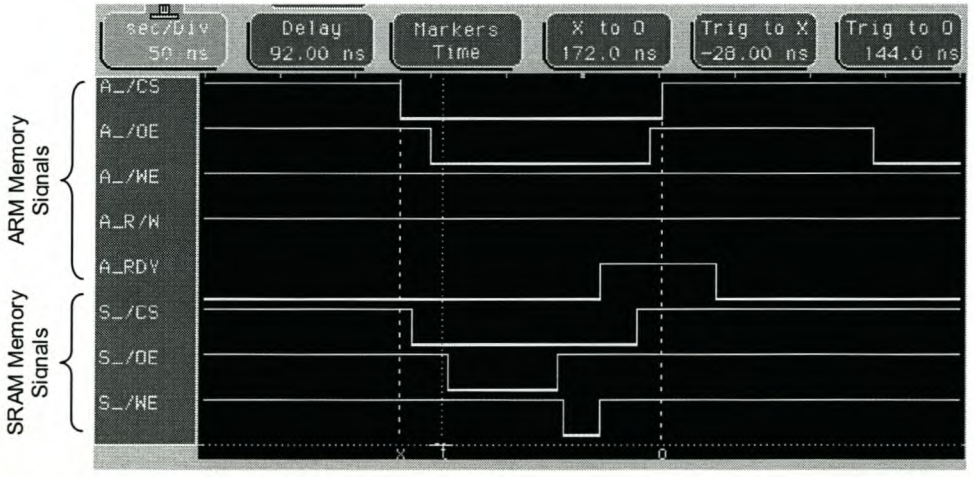

Figure 6-4: Extended Read Cycle

Initially, the EDAC system was tested by merely inverting one of the data or check-bits read from the SRAM into the FPGA. This test only proved that the data and check-bits were written to SRAM correctly and that the data correction circuitry worked. It did not, however, test the system completely. The extended read cycle writes the corrected data back to SRAM, and this write-back sequence still had to be tested. The following sequence was used to test the EDAC system thoroughly:

1. The processor writes a data word to a specific SRAM address with EDAC enabled, but one of the data bits written to the SRAM are first inverted within the FPGA logic. This causes the correct check-bits to be written to the check-bit SRAM, but the data written to SRAM has a single bit-flip error.
2. The processor reads the same address from SRAM with EDAC disabled. The data word containing the bit-flip error can be confirmed and the check-bits are not disturbed in this process.
3. The processor then performs the read again, with EDAC enabled. The EDAC system uses the original check-bits to correct the bit-flip error and writes the corrected data to SRAM (via the extended read cycle).
4. The processor reads the address again from SRAM with EDAC disabled. The corrected data word should be available, confirming that a successful write-back occurred.

6.3 The LVDS Channel

The LVDS channel was tested in a loop-back configuration. Data was written to the LVDS_TXD register. The data was then read back via the LVDS_RXD register and checked for consistency.

Initially, the system was tested using closed-loop synchronization, with the SYNC input of the serializer connected to the nLOCK output of the deserializer. However, the DS92LV1212 deserializer can also be used in an open-loop (random lock) system where it does not require sync patterns to achieve lock. This configuration was also tested, by keeping the SYNC input of the serializer high.

The logic analyser screen shots below, show the output data (R_{OUT}) when the hexadecimal value 0x01C is written to the LVDS_TXD register. The bus labelled "DATA" only show the lower 5 bits of R_{OUT}. The first screen shot (figure 6-5) shows the point of deserializer lock, as nLOCK goes low, the recovered clock (R_CLK) is activated and the R_{OUT} data lines are driven.

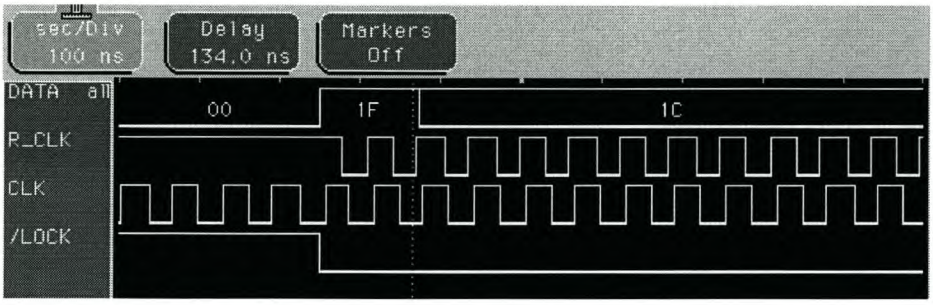

Figure 6-5: LVDS Deserializer Lock

Figure 6-6 shows the deserializer losing synchronization, nLOCK going high, and the loss of recovered clock (R_CLK). The last few data cycles show the incorrect value of 0x0C, indicating that the data was corrupted, due to the loss of synchronization.

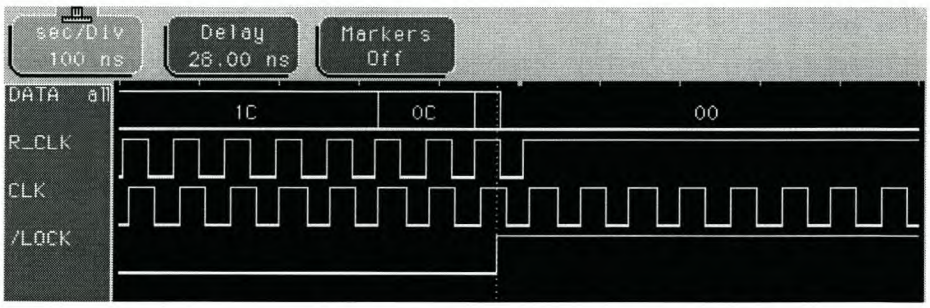

Figure 6-6: Deserializer Loosing Synchronization

Once synchronization is achieved again, nLOCK goes low, the recovered clock (R_CLK) starts again, and the correct data is received. This is shown in figure 6-7, below.

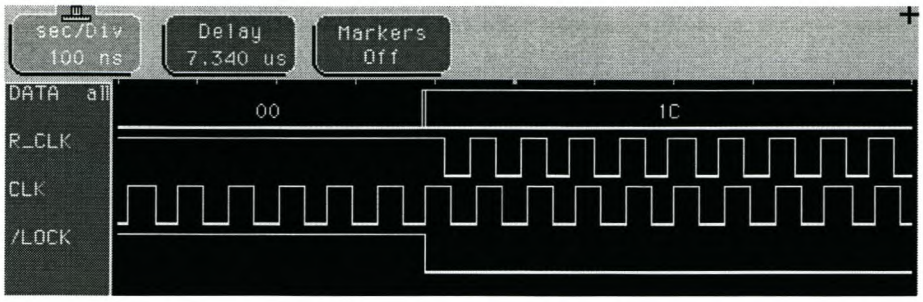

Figure 6-7: Deserializer Re-Synchronizes (open-loop)

6.4 The I²C Bus

The I²C slave devices were all configured and tested successfully, via test programs, using the software driver discussed in section 5.3.2.2.

The slow rate at which the SA-1110's GPIO pins can be toggled, limited the I²C transfers to approximately 40 kbps. Tests showed that the GPIO pins could only be toggled at a maximum rate of 187 kHz. For this reason, the FPGA should be used in a future design, especially if the I²C is to be used in fast-mode or high-speed mode for inter-subsystem communications.

CHAPTER 7

CONCLUSIONS AND RECOMMENDATIONS

7.1 Conclusions

The Intel StrongARM SA-1110 processor was identified as the most suitable ARM processor for use in a satellite OBC, according to the requirements set out in section 2.1. This can be summarised by the following two facts. Firstly, no 32-bit RISC processor, using the same size manufacturing process, can compete with the StrongARM's performance to power consumption ratio. Secondly, the SA-1110 processor's predecessor, the SA-1100, has a successful history of use in microsatellite and nanosatellite OBCs.

Conclusions drawn on the feasibility of using the SA-1110 in an OBC design are discussed in the following sections.

7.1.1 Reliability

Reliability in the space radiation environment is the main area of concern for an OBC based on the SA-1110. Fortunately, the 0.35µm manufacturing process of the device is fairly large, especially when its high performance and low current consumption are considered. Processors with similar performance are usually manufactured with smaller processes and are therefore more susceptible to radiation related errors[1].

Regrettably, no radiation test data, or history of use in space applications, is available for the SA-1110. The SA-1100 does, however, have a successful history of use in microsatellite and nanosatellite OBCs. The reliability of the SA-1110 should be similar as it is merely an upgraded version of the SA-1100 and is fabricated using the same manufacturing process.

[1] Compared to Hitachi SH7709S (0.25µm) and Samsung S3C2800 (0.18µm)

7.1.2 Power Consumption

The SA-1110 is a very power efficient device. This is very important for a microsatellite, which is typically solar powered. Its typical power consumption, of less than 350mW, is much lower than other processors with similar performance. For example, the Hitachi SH7709S consumes 880mW in typical applications. Even the most recently developed ARM processors cannot compete with the StrongARM: the Samsung S3C2800 consumes 625mW of power with similar computational performance (220 MIPS).

The power consumption of the SA-1110 can also be compared to older embedded processors, such as the Intel 80386EX processor, used in SUNSAT's second OBC [19]. The 80386 processor typically consumes 1.2 watts of power, which is very high in contrast to the 350mW of the SA-1110.

7.1.3 Processor Architecture and Performance

The StrongARM SA-1110 is a true 32-bit processor, with an external 32-bit data bus and internal 32-bit processing core. This gives it a great advantage over older 16-bit (or even 8-bit) processors, as far as data transfer rates and computational performance are concerned. The disadvantage of a wider external data bus, is that the complexity of the PCB layout increases. However, many new processors have a 64-bit architecture, giving them even better processing power at the expense of high design complexity. A 32-bit system can therefore be seen as a good compromise between performance and complexity.

The StrongARM's processing performance of 235 MIPS is very high compared to older embedded processors, such as the 80386EX. This will allow an OBC to carry out more complex control functions, and allow the other subsystems to be simplified, as in a C&DH control architecture.

The SA-1110 includes several useful on-chip peripherals that could be used in a satellite OBC, such as various serial ports, a real-time clock and watchdog timer. The processor also supports up to 28 external interrupt lines, which makes provision for a large number of peripheral devices.

7.1.4 Software Development

The operation of an OBC does not only depend on hardware, but on software too. For this reason, it is very important for the processor used in an OBC design to have good software support.

Due to the dominance of ARM processors in the embedded systems industry, the software support for these processors has grown tremendously over the past few years. A large number of software development suites and toolkits are available for ARM v4 compatible processors. The ARM-ELF GNU Cross Compiler (GCC) tool chain, used in this project, is one of the most popular development toolkits and is available free of charge. There are also many operating systems and real-time operating systems (RTOSs) available for the SA-1110 processor, including the popular ARM Linux.

7.2 Recommendations

The following sections highlight areas where further testing of the SA-1110 should be carried out. It also provides recommendations for improvements to the typical OBC design developed in this project.

7.2.1 Reliability testing

The reliability of the SA-1110 in the space environment is critical. Tests should be conducted on the reliability of the SA-1110 in environmental conditions similar to those that the OBC will be exposed to in space. Radiation tests on the processor should cover single event effects (SEE), and total ionising dose (TID). The total ionising dose will allow the life expectancy of the SA-1110 to be determined for a particular orbit. The SEE analysis will indicate the expected rate of hard and soft errors, such a SEL and SEU.

These tests should be conducted with the cache and buffer memories of the processor disabled and once again when enabled. This will allow the performance increases and risks associated with these memories to be analysed.

7.2.2 Processor Performance Tests

The manufacturers of the SA-1110 only provide the computational performance information of the processor when it is functioning optimally, with its cache and buffer memories enabled.

The performance, specified in Dhrystone MIPS (million instructions per second), does not take the external memory system into account, as the code is executed directly from the cache memories.

In order to obtain an accurate computational performance rating for the entire OBC system, the Dhrystone benchmark of the processor should be measured with the cache and buffer memories disabled.

7.2.3 Protected Memory System

The protected memory system has a large scope for improvement. The system used in this design is merely a simple example and is far from ideal. The protected memory was designed to only support full 32-bit data-words. This was done in order to simplify the EDAC system. This memory system should be improved to allow byte and half-word accesses by using the four byte-select lines from the processor.

Optimisation of the EDAC system should definitely be considered, as it will improve the overall performance of the OBC. Implementing the protected memory system using one of the SA-1110's standard static banks (without ready input) will allow the memory access times to be reduced dramatically. The complexity of the EDAC will however increase.

APPENDIX A

OBC PHYSICAL LAYOUT

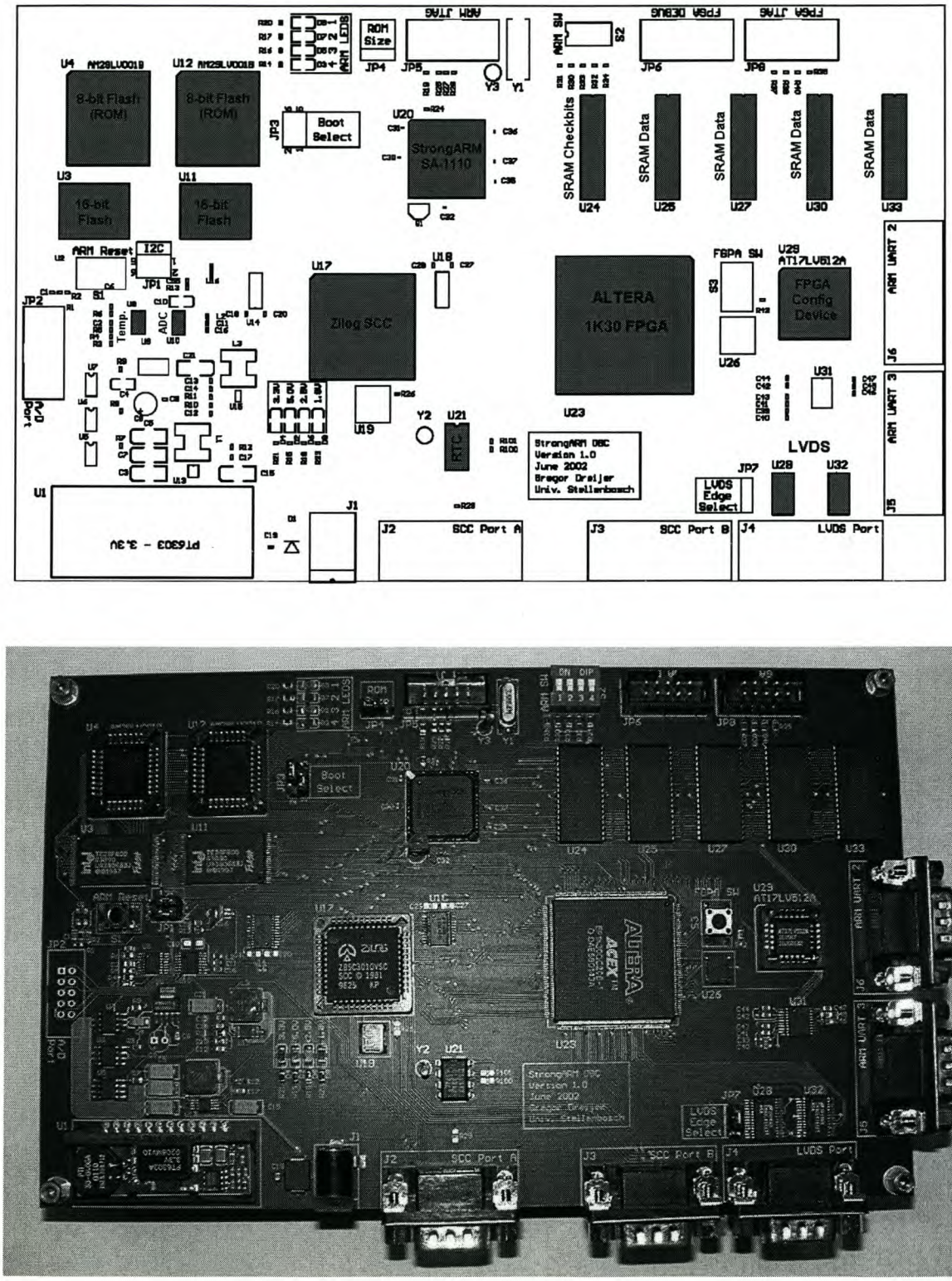

Figure A-1: Physical Layout of the Typical OBC System

APPENDIX B

CD-ROM DATA

The CD-ROM, below, contains the following data:

- VHDL Source Code, for the FPGA used in this design
- Program Source Code, in C, for the SA-1110 (utilities and test programs)
- Datasheets, for all of the components used in this design
- PCB Schematics of the OBC System

BIBLIOGRAPHY

[1] H. Grobler, "Aspects Affecting the Design of a Low Earth Orbit Satellite On-Board Computer", *Master's Thesis*, University of Stellenbosch, 2000.

[2] Intel StrongARM SA-1110 Microprocessor Developer's Manual, 2001

[3] Intel StrongARM SA-1110 Microprocessor Development Board User's Guide, 2000.

[4] A. Barnard, "Feasibility of Using an ARM Processor in a Micro Satellite On-Board Computer", *Master's Thesis*, University of Stellenbosch, 2001.

[5] Intel Application Note (AP-617) "Additional Flash Data Protection Using VPP, RP#, and WP#", 1998.

[6] P McCluskey, C O'Connor, and K Nathan, "Evaluating the Performance and Reliability of Embedded Computer Systems for Use in Industrial and Automotive Temperature Ranges", CALCE Electronic Products and Systems Centre, University of Maryland, 2001.

[7] J. Montanaro, "A 160-MHz, 32-b, 0.5-W CMOS RISC Microprocessor", *Digital Technical Journal*, vol. 9, Digital Equipment Corporation, 1997.

[8] C. Green, P. Gülzow, L. Johnson, K. Meinzer, and J Miller, "The Experimental IHU-2 Aboard P3D", *Proceedings of the 16th AMSAT Space Symposium, 1998.*

[9] N.L. Steenkamp, "Development of the On Board Computer Flight Software for SUNSAT 1", *Masters Thesis,* University of Stellenbosch, 1999.

[10] D. SEAL, ARM® Architecture Reference Manual, 2nd Edition, Addison-Wesley, Harlow, UK, 2001.

[11] ARM Developer Suite Version 1.0.1, Developer Guide, ARM Limited, 2000.

[12] M.V. O'Bryan, K.A. La Bel, R.A. Reed, J.W. Howard, J.L. Barth, C.M. Seidleck, P.W. Marshall, C.J. Marshall, H.S. Kim, D.K. Hawkins, M.A. Carts, and K.E. Forslund, "Recent Radiation Damage and Single Event Effect Results for Microelectronics", 1999.

[13] T. Lin, G. Lyons, and F. Schapfel, "Protecting Your Data With the IDT49C465 32-Bit Flow-thruEDC Unit", Application Note (AN-64), Integrated Device Technology Inc., 1996.

[14] S.M. Parkes, "High-Speed, Low-Power, Excellent EMC: LVDS For On-Board Data Handling", University of Dundee, 19xx.

[15] The I^2C-Bus Specification Version 2.1, Philips Semiconductors, 2000.

[16] R.C. Webb, "The Commercial and Military Satellite Survivability Crisis", *Defense Electronics, vol.* 27, no. 8 (August 1995), pp. 21-25.

[17] H. Eda, and T. Shintoh, "ARM CPU Core Dominates Mobile Market", *Nikkei Electronics Asia*, April 2002 Issue.

[18] C. Underwood, "SNAP-1: A Low Cost Modular COTS-Based Nano-Satellite – Design, Construction, Launch and Early Operations Phase", Proceedings of 15th AIAA/USU Conference on Small Satellites, (SSC01-V-1b), 2001.

[19] H.J. Gouws, "Design of an On-Board Computer for the SUNSAT Microsatellite", *Master's Thesis*, University of Stellenbosch, 1996.